Treball de Fi de Grau

Grau en Enginyeria en Tecnologies Industrials

Construcción de un micro-brazo articulado: Parte electrónica

MEMÒRIA

Autor: Francisco Javier Estévez Afonso
Director: Juan Manuel Moreno Eguílaz

Convocatòria: Octubre 2016

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resumen

Este proyecto tiene como objeto de estudio un micro brazo robótico basado en una paletizadora de la empresa ABB. El estudio describe el proceso de diseño e implementación de la electrónica necesaria para su correcto control del movimiento a través de una aplicación móvil.

El objetivo principal del proyecto es la implementación en el robot de un hardware y un software capaz de dotar de movimiento al brazo, a la vez que se controla un electroimán para transportar elementos ferromagnéticos, por wifi y autónomamente. Para poder conseguir dicho objetivo se utiliza la plataforma Blynk y su app para Play Store de Android o App Store de Apple.

Para ello se analiza y se selecciona el hardware utilizado y se diseña y se describe el software programado en el lenguaje JavaScript. Se aplican conocimientos de cálculo, geometría, informática y electrónica.

La conclusión del proyecto es un robot totalmente operativo, controlado por un dispositivo móvil tipo teléfono inteligente y una guía completa del montaje y funcionalidad de la electrónica con opciones de mejora.

Sumario

1. Glosario

GPIO – *General Purpose Input/Output*, Entrada/Salida de Propósito General. Pin genérico en un chip, cuyo comportamiento se puede controlar por el usuario en tiempo de ejecución.

Hardware – Vertiente de la electrónica que engloba las partes físicas de un sistema informático.

LED – *Light-Emitting Diode*, Diodo emisor de luz.

Librería – Conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.

MOSFET – *Metal-oxide-semiconductor Field-effect transistor,* Transistor de efecto de campo metal-óxido-semiconductor. Transistor utilizado para amplificar o conmutar señales electrónicas.

Node.js – entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. El ecosistema de paquetes de Node.js, npm, es el ecosistema más grande de librerías de código abierto en el mundo.

PBB – *Polybrominated Biphenyls*, Polibromobifenilos. Grupo de compuestos químicos del bromo. Es un retardante nocivo para la salud humana y el medio ambiente.

PBDE – Los polibromodifenil éteres son una clase de compuestos bromados de extenso uso como retardantes de llama en plásticos y espumas.

PLA – *Polylactic Acid, Á*cido Poliláctico. Polímero termoplástico y biodegradable que se utiliza como material del modelo en la técnica de conformado de la estructura del brazo robótico.

PWM – *Pulse Width Modulation*, Modulación de anchura de pulso.

RAM – *Random Access Memory*, Memoria de acceso aleatorio

Rlogin – *Remote Login*, Acceso remoto. Aplicación TCP/IP que comienza una sesión de terminal remoto sobre el anfitrión especificado como host.

RoHS – *Restriction of Hazardous Substances,* Restricción de Sustancias Peligrosas. Normativa referente a sustancias peligrosas encontradas en dispositivos eléctricos y electrónicos.

SBC – *Single Board Computer,* Ordenador de placa reducida.

SD – *Secure Digital Card.* Tarjeta de memoria no-volátil.

Software – Vertiente intangible de la electrónica que como función principal diseña el código con las instrucciones que ejecutaran los microcontroladores.

SPI – *Serial Peripheral Interface Bus*. Bus estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos.

TCP – *Transmission Control Protocol*, Protocolo de Control de Transmisión. Uno de los protocolos fundamentales en Internet que garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron.

Telnet – *Telecommunication Network,* Red de Telecomunicaciones. Protocolo de red que nos permite viajar a otra máquina para manejarla remotamente como si estuviéramos sentados delante de ella

USB – *Universal Serial Bus*, Bus Serie Universal. Bus estándar industrial utilizado por cables, conectores y protocolos para comunicar, conectar y alimentar componentes electrónicos, periféricos o computadores.

V_{th} – Tensión de umbral del transistor, tensión que si se sobrepasa el mosfet abandona la zona de corte y entra en la región lineal o de saturación.

VNC – *Virtual Network Computing*, Computación Virtual en Red).

WEEE – *Waste Electrical and Electronic Equipment*, Directiva de Residuos de Aparatos Eléctricos y Electrónicos. Ley en vigor desde el 13 de agosto del 2005 en todo el ámbito de la Unión Europea.

2. Prefacio

2.1. Origen del proyecto

El origen del proyecto está basado en la idea del profesor Manuel Moreno Eguílaz de diseñar y construir un brazo robótico para que en un futuro alumnos de la ETSEIB puedan desarrollar sus habilidades y pongan en práctica sus conocimientos estudiando el modelo y programando la electrónica en asignaturas como Proyecto I ó Proyecto II.

El proyecto se ha dividido en dos partes independientes y por tanto dos TFGs. La primera es la parte mecánica, donde se diseña y se construye la estructura del robot [1]; y la segunda es la parte electrónica, donde a partir de unos servos y una placa computadora se consigue el movimiento del mismo.

Esta memoria hace referencia a la segunda parte definida y, por tanto, las posteriores líneas tratarán sobre los elementos electrónicos y la programación utilizada para hacer posible el movimiento del brazo.

2.2. Motivación

La motivación principal del proyecto ha sido poder aplicar los conocimientos adquiridos en varias de las asignaturas del 'Grau en Enginyeria en Tecnologies Industrials', impartido en la ETSEIB, dado el carácter multidisciplinar del proyecto.

En especial, la parte más enriquecedora a nivel académico es la interconexión de los diferentes sistemas a nivel de comunicación, donde a partir de una aplicación móvil (app) para teléfonos inteligentes se puede controlar por completo las funcionalidades del robot.

2.3. Requerimientos previos

Para poder completar el proyecto con éxito, aparte de aplicar los conocimientos adquiridos a lo largo del grado, también es necesario tener conocimientos sobre JavaScript, lenguaje en el cual se ha programado la placa computadora. También es necesario aprender y conocer cómo funciona un microcomputador y la aplicación Blynk. Por último, hay que adquirir conocimientos sobre pulsos modulados en anchura para poder mover los servomotores.

3. Introducción

3.1. Objetivos del proyecto

El principal objetivo del proyecto es diseñar y programar la electrónica encargada del movimiento de un prototipo de un modelo de "MicroArm", parecido al que se comercializa en el mercado por parte de la empresa UFactory [2].

Otros objetivos secundarios son:

- Controlar todos los movimientos y funcionalidades del robot remotamente por wifi y sin cables mediante una App móvil.
- Dotar al robot de una batería para que sea totalmente autónomo y que se pueda utilizar en cualquier lugar donde haya wifi.
- Procurar, teniendo en cuenta que se trata de un prototipo, minimizar los costes.
- Buscar todos los proveedores próximos por si se quiere repetir la construcción del modelo.

3.2. Alcance del proyecto

Este proyecto engloba la implementación y programación de la electrónica necesaria para el correcto movimiento y funcionamiento del brazo robótico, cuyo control se realizará con un teléfono inteligente mediante wifi. Para ello se utilizarán unas librerías de código abierto para controlar las dos placas que se utilizan en el proyecto. En ningún momento se modificarán dichas librerías para adaptarlas a las necesidades del proyecto.

El control de los servomotores y el electroimán se realizará en lazo abierto, y en el caso de los servomotores mediante consignas de ángulos girados.

Por último, la programación realizada no incluirá ningún tipo de algoritmo de control de movimiento.

4. Desarrollo del Proyecto

4.1. Introducción

Para poder cumplir con los objetivos previamente descritos hará falta un dispositivo móvil inteligente (Smartphone), donde se instalará la aplicación Blynk [3]. La aplicación, con la cual se controlará el movimiento del robot y la funcionalidad del electroimán, se comunicará mediante wifi con un microcomputador (Raspberry Pi 3), que a su vez se comunicará con un módulo controlador (PCA9685) donde estarán los servos conectados. Al mismo tiempo, el electroimán del cabezal estará conectado al microcomputador.

Una vez que todos los componentes están conectados se crea un script con el lenguaje de programación Javascript donde, además de establecer la comunicación entre el Smartphone y Raspberry Pi, se programa un software para que tanto los servos como el electroimán funcionen correctamente.

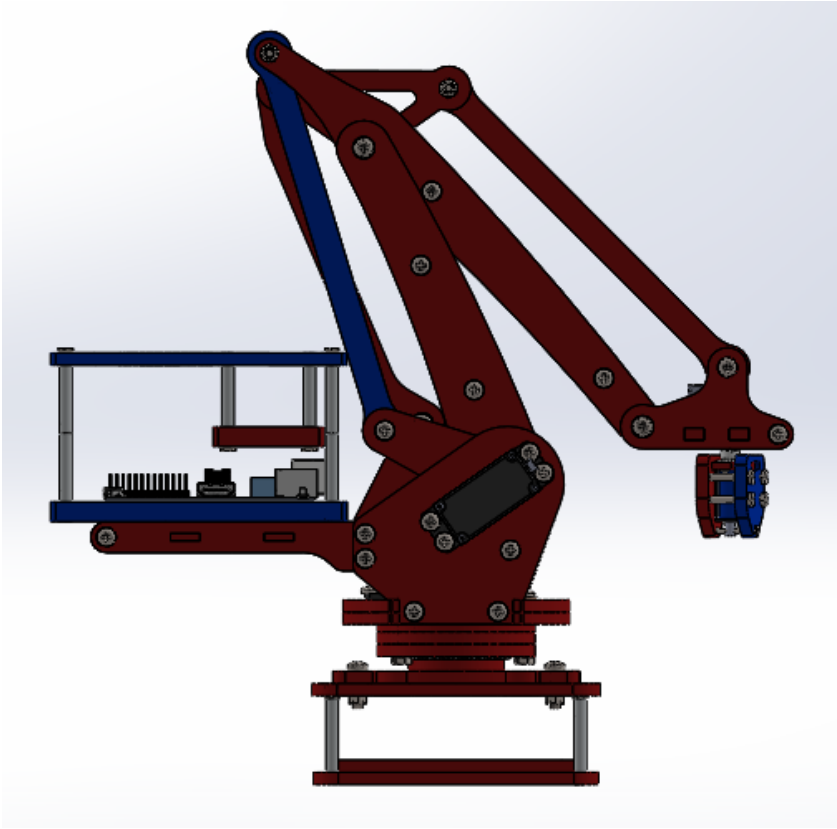


Fig. 4.1. Imagen brazo robótico. Fuente: TFG Guillem [1]

4.2. Hardware

4.2.1. Raspberry Pi 3

En este apartado se exponen las diferentes opciones de hardware que se han contemplado para el control del brazo robótico.

Actualmente en el mercado existe una gran variedad de microcontroladores, de los cuales, se han seleccionado los cuatro más conocidos. Entre ellos cabe comentar que hay dos subgrupos; Wipy 2.0 y Arduino UNO son microcontroladores mientras que BeagleBone Black y Raspberry Pi 3 son microcomputadores.

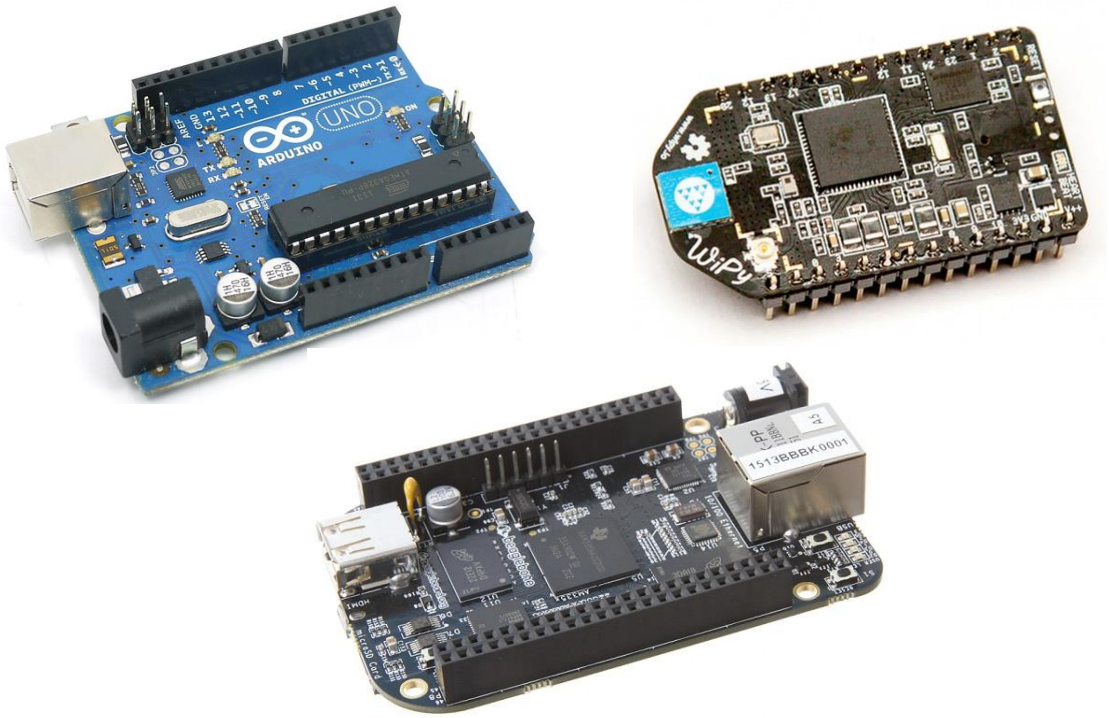


Fig. 4.2. Imagen Arduino Uno, Wipy 2.0 y BeagleBone Black. Fuente: Google.es

A continuación se incluye una tabla con las principales características de cada una de las placas electrónicas mencionadas.

	Wipy 2.0	Arduino UNO	BeagleBone Black	RaspberryPi 3
Procesador	ESP32	ATMega328	AM33x	ARM Cortex-A53
Velocidad	32 KHz	16 MHz	1 GHz	1,2 GHz
RAM	512 Kb	2 Kb	512 Mb	1Gb
USB	-	-	1	4
Audio/Video	-	-	HDMI	HDMI
GPIO Digitales	24	14	66	27
Entradas PWM	4	6	8	4
Internet	Wifi	Ethernet	Ethernet	Wifi/Ethernet
Sist. Operativo	-	-	Android, Linux, Windows	Linux, Windows
Precio	19,95 €	22,45 €	33,75 €	49,25 €

Tabla 4.1. Placas microcontroladoras. Fuente: Propia

Las cuatro placas son capaces de controlar el movimiento de los servos y del electroimán, pero para poder crear un robot autónomo y portátil sin necesidad de cables se necesita controlarlo por wifi, bluetooth o cualquier otro sistema inalámbrico de comunicaciones. Por tanto, desde un principio BeagleBone Black y Arduino UNO han sido descartadas, ya que para poder controlarlas habría que añadirle un módulo wifi.

Comparar un pequeño microcontrolador como Wipy 2.0 con un microcomputador como Raspberry Pi 3 no se puede, ya que han sido diseñados con un propósito diferente. Raspberry Pi 3 realmente es un ordenador del tamaño de una tarjeta de crédito y por lo tanto, tiene una capacidad de cálculo y velocidad mucho mayor, al mismo tiempo que puede ser utilizado para un mayor número de aplicaciones dada su versatilidad. No obstante, para el proyecto en cuestión Wipy 2.0 cumplía con los requisitos necesarios y dado su menor precio fue la opción escogida.

Por desgracia la placa solo se podía adquirir vía web y la empresa no pudo cumplir con el plazo de entrega. Ante tal situación se decidió optar por Raspberry Pi 3, que aun siendo más cara, también es más conocida a nivel internacional.

La decisión final por tanto ha sido utilizar una Rasberry Pi 3 con el sistema operativo basado en Linux, Raspbian. Será aquí donde se ejecutará el programa que hará posible el movimiento del brazo robótico.

Fig. 4.3. Imagen Raspberry Pi 3. Fuente: Google.es

4.2.2. PCA9685

Esta placa electrónica es un módulo controlador de 16 canales que se comunica mediante bus I2C.

EL controlador PCA9685 [4] fue diseñado para controlar leds mediante señales PWM, pero también nos permite controlar servos, ya que estos también se controlan por PWM, aplicación que actualmente es muy usada.

El módulo controlador de servos PCA9685 tiene la placa diseñada para el control de servos, teniendo los pines en el orden correcto para simplemente conectar los servomotores, además de una bornera para la alimentación de los servos y conectores para la alimentación de la parte lógica, junto con los pines I2C para comunicarse con la Raspberry.

EL PCA9685 nos permite controlar individualmente hasta 16 salidas PWM, con 12 bits de resolución y una frecuencia máxima de 1600Hz.

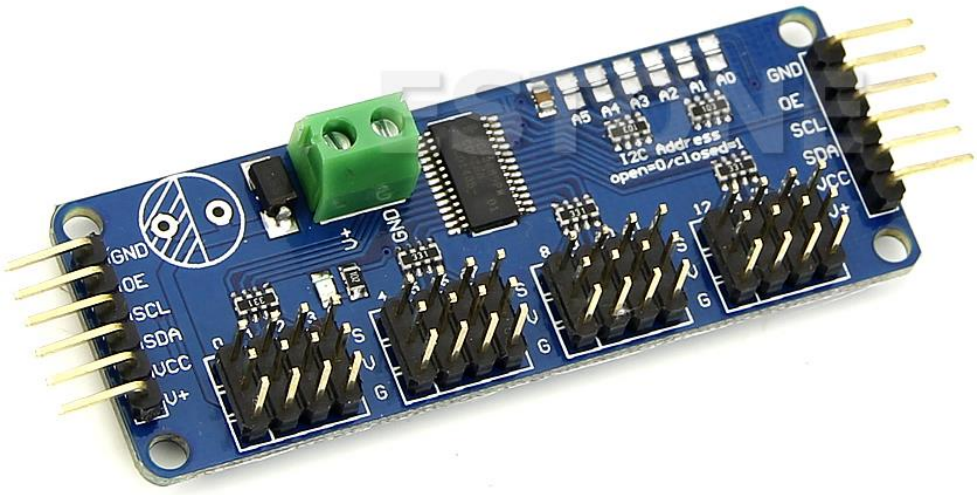


Fig. 4.4. Imagen Placa PCA9685. Fuente: Google.es

4.2.3. Servomotores

Los encargados de hacer posible el movimiento del brazo robótico son los servomotores, también denominados servos por abuso del lenguaje. Un servo es muy similar a un motor de corriente continua, con la diferencia de que es capaz de situarse en cualquier posición angular y mantenerse de forma indefinida.

En particular, el micro-brazo contiene 4 servomotores; dos de 180 grados, que permiten el movimiento biela-manivela, uno también de 180 grados para la rotación del cabezal y por último uno de 360 grados, que permite la rotación respecto el eje vertical. Este último es distinto a los otros, ya que es de rotación continua.

Para la elección de los servomotores se tuvo en cuenta tanto el precio como las necesidades mecánicas y geométricas. También influyó el objetivo de buscar proveedores cercanos en el caso de que se quiera volver a reproducir el proyecto.

Los dos servomotores más críticos mecánicamente hablando son los que permiten el movimiento del brazo. Para poder elegir ambos se ha hecho un breve estudio sobre la dinámica del brazo y el cabezal.

Para obtener la masa tanto del brazo como del cabezal se utiliza la siguiente ecuación, teniendo en cuenta que la densidad del PLA, material que conforma el brazo, es de 1,23 g/cm3.

$$m = V \cdot \rho \qquad \text{(Ec. 4.1).}$$

A continuación se muestra una tabla con los elementos que conforman el brazo y cabezal y el peso que cada servo tendrá que soportar.

	Volumen [Cm^3]	Masa [g]
Brazo	151,18	185,95
Cabezal	27,12	33,36
Servomotor (cabezal)	-	9,92
Electroimán	-	10
Tornillería	-	57

Tabla 4.2. Peso de elementos del brazo. Fuente: Propia

El brazo totalmente extendido tiene una longitud aproximada de 30 cm. En esta configuración obtenemos la posición crítica en la que los servos padecerán un esfuerzo máximo. Es por ello que mediante la ecuación del par obtenemos los siguientes resultados.

$$\Gamma = F \cdot d \quad \text{(Ec. 4.2).}$$

$$\Gamma_{cabezal} = 0{,}05328\ kg \cdot 30\ cm = 1{,}6\ kgf \cdot cm \quad \text{(Ec. 4.3).}$$

$$\Gamma_{brazo} = 0{,}243\ kg \cdot 15\ cm = 3{,}65\ kgf \cdot cm \quad \text{(Ec. 4.4).}$$

$$\Gamma_{carga} = 0{,}1\ kg \cdot 30\ cm = 3\ kgf \cdot cm \quad \text{(Ec. 4.5).}$$

$$\Gamma_{total} = \Gamma_{cabezal} + \Gamma_{brazo} + \Gamma_{carga} = 8{,}25\ kgf \cdot cm \quad \text{(Ec. 4.6).}$$

Aproximando cálculos, aplicamos el peso del electroimán, el servo y el cabezal en el extremo del brazo (30 cm). Por contra, el peso del brazo y la tornillería la aplicamos en su centro de masas, que sería el centro (15 cm).

De estos cálculos extraemos la conclusión de que los servomotores de 180º del movimiento biela-manivela deben tener un par máximo mayor a 8,25 kgf·cm.

A continuación se muestra una tabla con los tres diferentes tipos de servomotores utilizados y sus características principales.

	Biela-Manivela	Cabezal	Rotación (360º)
Modelo	MG995	SM-S2309S	SM-S4303R
Par (4,8 V) [kgf·cm]	8,5	1,1	3,3
Dimensiones [mm]	40,7 x 19,7 x 42,9	22,9 x 12,3 x 22,2	41,3 x 20,7 x 40,2
Velocidad (4,8 V)	0.2s/60º	0,11s/60º	43 rpm
Peso [g]	55	10	41
Voltaje operativo [V]	4,8 a 7,2	4,8 a 7,2	4,8 a 7,2

Tabla 4.3. Servomotores del brazo robótico. Fuente: Propia

Como se puede observar los servos Biela-Manivela tiene un par máximo de 8,5 kgf·cm, mayor a 8,25 kgf·cm que debe soportar. El servo del cabezal tiene que tener unas dimensiones reducidas por la geometría del cabezal y no debe tener un par máximo elevado, ya que no es necesario para su función. Por último, el servo de rotación continua tampoco debe tener un par muy elevado, ya que su movimiento es alrededor del eje vertical al igual que el servo del cabezal.

Funcionamiento del servo. Control PWM

La modulación por anchura de pulso, PWM (Pulse Width Modulation), es uno de los sistemas más empleados para el control de servos. Este sistema consiste en generar una onda cuadrada en la que se varía el tiempo que el pulso está a nivel alto, manteniendo el mismo período, con el objetivo de modificar la posición del servo según se desee.

A continuación se muestra una figura donde se puede observar una señal PWM y sus características principales.

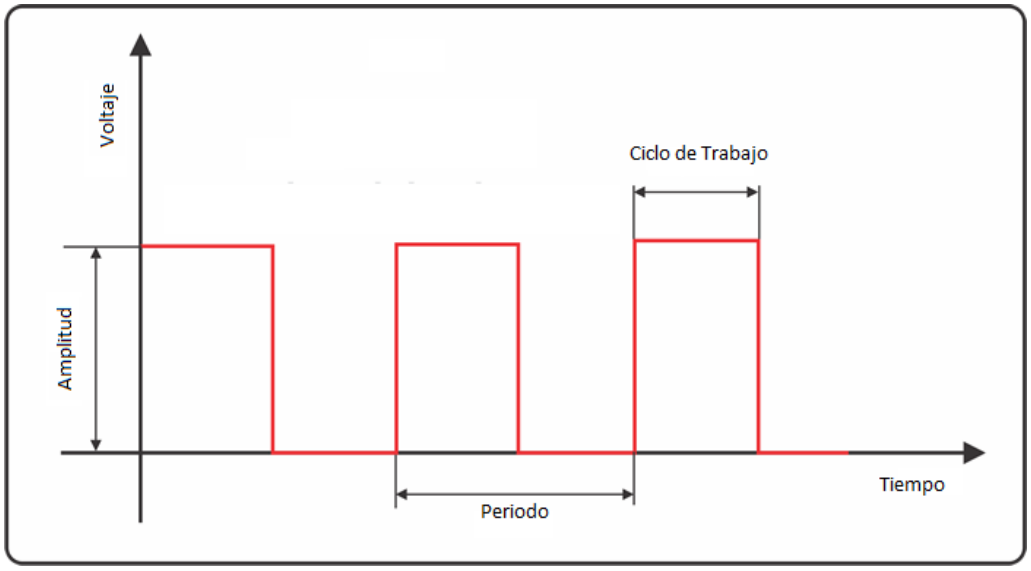


Fig. 4.5. Gráfico onda PWM. Fuente: Propia

En el caso de los servos utilizados en el proyecto, a excepción del servo de rotación continua que funciona de distinta forma, sus características se muestran en la siguiente tabla.

	Servomotor
Periodo [ms]	20
Amplitud [V]	5
T_{on} 0º [ms]	1
T_{on} 90º [ms]	1,5
T_{on} 180º [ms]	2

Tabla 4.4. Características PWM del servomotor. Fuente: Propia

El servo de rotación continua, a diferencia de los otros, no tiene la capacidad de escoger una posición concreta cambiando el ciclo de trabajo del PWM, sino que lo que varía al cambiar esta característica es la velocidad y sentido de giro. En el caso de este tipo de servo, al aplicarle un ciclo de trabajo (T_{on}) de 1,5 ms, estaría en reposo. En cambio, si dicho ciclo de trabajo es de 1 ms ó 2 ms, gira a máxima velocidad en un sentido y en el otro respectivamente.

Para poder controlar los servomotores se utiliza la placa PCA9685 de 12 bits. Con estos 12 bits se tienen 4096 valores distintos y en este formato se enviará la información a la placa PCA9685. Mediante la siguiente fórmula se relaciona el pulso y la frecuencia con el valor que se enviará a la placa controladora.

$$n_{bits} = Pulso_{ms} \cdot f_{kHz} \cdot 4096 \qquad \text{(Ec. 4.7)}$$

4.2.4. Electroimán

El electroimán escogido es un solenoide de la marca Cebek y modelo C-6092. Su funcionamiento es muy sencillo: al conectarlo a una fuente de corriente continua el émbolo es atraído hacia su interior y mediante un campo magnético atrae elementos ferromagnéticos.

Sus características principales se muestran en la siguiente tabla.

	Electroimán C-6092
Voltaje [V]	12
Potencia [W]	1,1
Corriente [mA]	92
Peso [g]	10
Fuerza máx. [N]	3,92

Tabla 4.5. Características del electroimán. Fuente: Propia

Fig. 4.6. Imagen Electroimán C-6092. Fuente: Google.es

4.2.5. Batería

Para alimentar todo el sistema electrónico y a los servomotores se necesita una fuente de energía eléctrica. Esta función la desempeñará una batería recargable, ya que además de dotar al robot de autonomía, una vez consumida toda la energía almacenada por ésta, conectándola a la red se puede reutilizar ilimitadamente.

Para ello se ha elegido un Power Bank de la marca Ideus y modelo PB6600. Esta batería externa ha sido diseñada para la recarga de móviles, pero cumple con los requisitos necesarios para el proyecto. Tiene dos salidas USB para alimentar la Raspberry Pi y los servos.

En la tabla siguiente aparecen las características principales de la batería.

	Batería PB6600
Voltaje [V]	5
Capacidad [mAh]	6600
Salidas USB	2
Corriente [A]	1 – 2
Peso [g]	240
Composición	Li-Ion

Tabla 4.6. Características de la batería.
Fuente: Propia

Fig. 4.7. Imagen Batería Ideus 6600.
Fuente: Google.es

4.3. Conexión

En este apartado se explica cómo deben ir conectados los diferentes sistemas electrónicos del brazo robótico. La conexión se dividirá en dos partes; por un lado, la conexión de los servomotores y el módulo PCA9685 y por otro el circuito necesario para el correcto funcionamiento del electroimán.

4.3.1. PCA9685

Para poder controlar los 4 servos se necesita el módulo PCA9685 de 16 canales, ya que la Raspberry Pi no puede controlarlos a la vez. También es recomendable usar este módulo para alimentar los servos independientemente, ya que la Raspberry Pi tiene limitaciones en cuanto a la intensidad de sus salidas.

La comunicación entre la Raspberry Pi y el módulo controlador PCA9685 se hace mediante bus I^2C.

I^2C es un bus serie síncrono de datos y se utiliza principalmente para la comunicación entre diferentes partes de un circuito, por ejemplo, entre un controlador y circuitos periféricos integrados.

A continuación se muestra una figura detallada con la conexión de ambas placas y los servos realizada mediante el programa Fritzing [5].

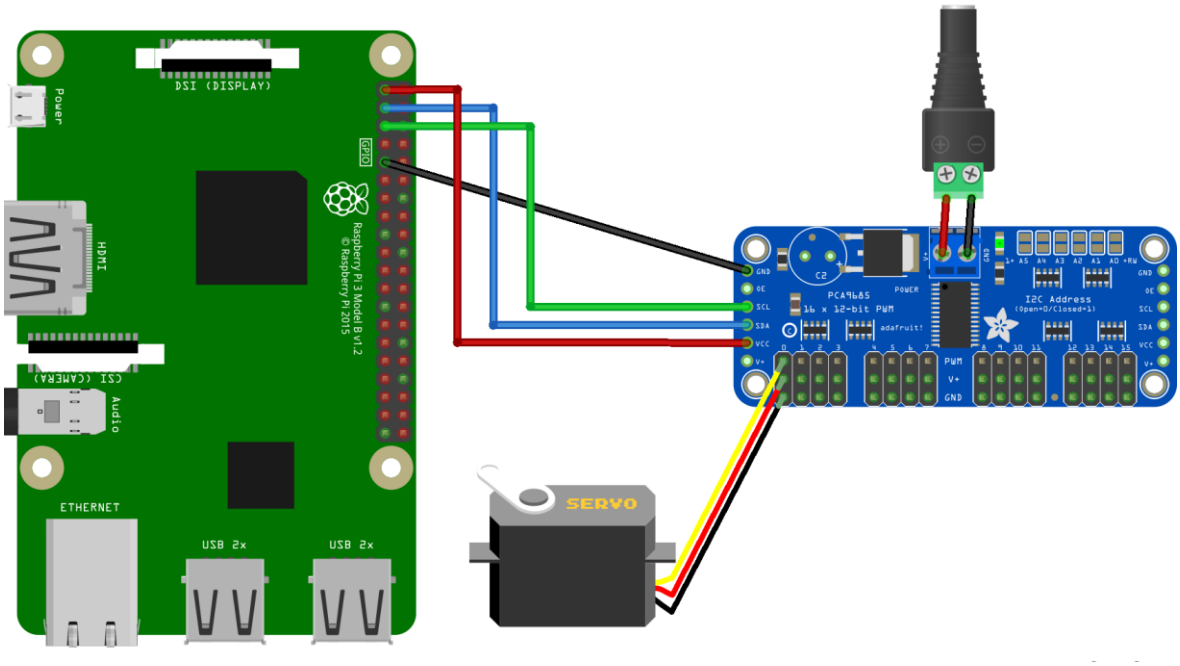


Fig. 4.8. Esquema cableado servomotores mediante programa Fritzing. Fuente: Propia

Para que la comunicación sea posible se tienen que conectar los pines SCL (línea de reloj) y SDA (línea de datos) de ambas placas. También hay que alimentar a 3,3V la placa controladora de 16 canales (pin VCC) y conectar las masas de ambas placas (GND).

Cada servo hay que colocarlo en uno de los 16 canales, conectando la señal PWM de color naranja, blanco o amarillo; alimentación de color rojo y masa de color negro o marrón en su pin correspondiente (PWM, V+ y GND).

Por último, hay que alimentar tanto la Raspberry Pi como los servos mediante un cable USB conectado a la batería. En el caso de los servos se hace mediante la placa PCA9685, donde el cable negro del USB debe ir al pin GND y el rojo al pin V+.

4.3.2. Electroimán

Para el control y el correcto funcionamiento del electroimán se ha tenido que diseñar un circuito adicional soldado sobre una placa de topos. Dicho circuito lo componen un transistor MOSFET de canal n de enriquecimiento, un módulo convertidor CC/CC, un diodo y un par de resistencias.

MOSFET

El uso de este componente es debido a que la tensión de los pines de la Raspberry Pi es de 3,3V y el electroimán debe alimentarse a 12V.

El dispositivo contiene tres terminales denominados fuente (S, Source), drenador (D, Drain) y puerta (G, Gate) [6].

Su funcionamiento se puede dividir en tres regiones de operación.

- Si $V_{GS} < V_{th}$ el transistor se comporta como un interruptor abierto y por lo tanto no hay conducción entre la fuente y el drenador.
- Si $V_{GS} > V_{th}$ y $V_{DS} < (V_{GS} - V_{th})$ el transistor se comporta como una resistencia controlada por la tensión de puerta.
- Si $V_{GS} > V_{th}$ y $V_{DS} > (V_{GS} - V_{th})$ el transistor se comporta como una fuente de corriente controlada por la tensión V_{GS}.

Debido a estos comportamientos, cuando mediante la Raspberry Pi se genere un estado lógico alto, V_{GS} será mayor que V_{th} y por lo tanto habrá conducción entre el drenador y el surtidor y el electroimán se activará. En caso contrario no habrá circulación y el electroimán se desactivará.

El transistor MOSFET elegido ha sido el IRL2505. Se ha escogido este transistor en particular ya que las salidas PWM de la Raspberry Pi son de 3,3V y por lo tanto la tensión umbral (2V) tiene que ser menor para utilizarlo como un interruptor. Tanto la tensión entre drenador y surtidor como la corriente que pasa por el drenador se han sobredimensionado teniendo en cuenta que el electroimán se alimenta a 12V y consume una corriente máxima de 92 miliamperios. A continuación se muestran sus características principales.

	Mosfet IRL2505
V_{th} [V]	1-2
Enriquecimiento	Canal N
V_{DSS} [V]	55
I_D [A]	104
$R_{DS(on)}$ [mΩ]	8

Tabla 4.7. Características del Mosfet. Fuente: Propia

Módulo convertidor CC/CC

El electroimán debe alimentarse con una tensión de 12V. Tanto las salidas USB de la batería como de la Raspberry Pi son de 5V, insuficientes para su correcto funcionamiento. Para solventar el problema se añade al circuito un módulo convertidor cuya función es elevar la tensión con la que alimentamos el electroimán de 5V a 12V.

El módulo escogido es de la marca Traco Power y modelo TMR-0512. Es un convertidor CC-CC de 2W aislado, cuyas características principales se exponen a continuación.

Pin	Función
1	$-V_{in}$ (GND)
2	$+V_{in}$ (Vcc)
3	Remote On/Off
4	-
5	$+V_{out}$
6	$-V_{out}$
7	-

Tabla 4.8. Función de los pines de TMR-0512.
Fuente: Propia

	Convertidor TMR-0512
V_{In} [V]	4,5 – 9
V_{Out} [V]	12
Potencia [W]	2
I_{max} [mA]	645

Tabla 4.9. Características TMR-0512.
Fuente: Propia

Diodo

La función de este componente electrónico es proteger el transistor de los picos de tensión que se producen cuando se conmuta la carga.

El diodo escogido ha sido el 1N4004 [7], cuyas características principales se muestran en la siguiente tabla.

	Diodo 1N4004
V_{RRM} [V]	400
V_{RWM} [V]	280
V_R [V]	400
V_F [V]	1,1

Tabla 4.10.Características del diodo.
Fuente: Propia

Resistencias

Por último, al circuito se le añaden dos resistencias de 100Ω y 100kΩ. Su misión es limitar la corriente de puerta y descargar la capacidad de la puerta.

Circuito

Para poder conectar todos los componentes se ha utilizado una pequeña placa de topos donde se han soldado cada uno de ellos.

A continuación se expone una imagen creada mediante el programa Fritzing, donde se aprecia el circuito necesario para el funcionamiento del electroimán.

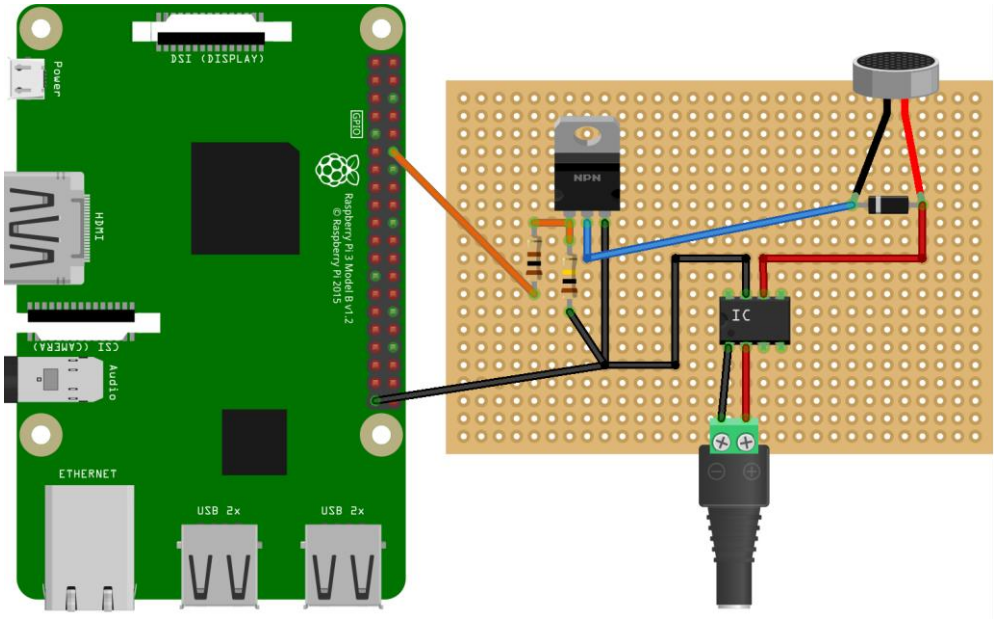


Fig. 4.9. Esquema cableado electroimán mediante programa Fritzing. Fuente: Propia

Para la conexión se deberán tener varios puntos en cuenta:

- Entre el terminal G del transistor (puerta) y la salida de la Raspberry se añade una resistencia de 100Ω. Entre G y S (surtidor) se debe poner una resistencia de 100kΩ.
- El terminal S debe estar conectado a un pin GND de la Raspberry Pi y al pin 6 del convertidor.
- El terminal D (drenador) debe estar conectado a uno de los terminales del electroimán. El otro terminal del electroimán se debe conectar al pin 5. Entre ambos terminales del solenoide se debe colocar el diodo en dirección hacia el convertidor.
- El convertidor se alimenta con un USB conectado a la Raspberry Pi, donde el cable negro irá al pin 1 y el cable rojo al pin 2.

4.4. Software

Una vez escogido todos los componentes y después de haberlos conectado correctamente en este capítulo se procederá a explicar detalladamente la programación realizada y el software utilizado.

4.4.1. Raspbian

En primer lugar, se debe instalar un sistema operativo en la Raspberry Pi 3. Existe una gran variedad de sistemas operativos de entre los cuales se han descartado los que no fueran distribuciones de Linux. De esta lista se han seleccionado los dos que se utilizan como PC o como servidor y que a la vez son los más descargados y utilizados en Raspberry Pi 3. Estos dos son Raspbian [8] basado en Debian y Pidora [9] que es un remix de Fedora.

Raspbian OS es la distribución por excelencia para la Raspberry Pi. Es la más completa y optimizada de las existentes, por eso cuenta con apoyo oficial. Raspbian OS se basa en la potente distro Debian Wheezy (Debian 7.0) optimizando el código de ésta para la SBC Raspberry Pi. La distribución es ligera para moverse ágilmente en el hardware de la Raspberry Pi, con un entorno de escritorio LXDE y Midori como navegador web predeterminado.

Fig. 4.10. Logo Raspbian. Fuente: Google.es

Pidora es básicamente una distribución Linux Fedora especialmente optimizada para funcionar en ARM. Por el resto de características es similar a Fedora, la hermana pequeña de Red Hat, y mantenida por los mismos desarrolladores de esta comunidad libre. También está reconocida oficialmente por la comunidad de Raspberry Pi.

Fig. 4.11. Logo Pidora. Fuente: Google.es

Finalmente, el sistema operativo escogido ha sido Raspbian, ya que es el sistema más completo, intuitivo y el cual cuenta con más soporte y actualizaciones.

Una vez escogido el sistema operativo se procede a su instalación. Para ello se necesitará una tarjeta SD de como mínimo 4GB de capacidad, siendo recomendable usar una con mayor capacidad.

Primeramente, se debe descargar el archivo .zip de la página oficial [7] en un PC. Una vez descargado se debe extraer el archivo comprimido mediante un programa, por ejemplo 7-Zip. Una vez extraído se dispondrá de un archivo .img.

A continuación, mediante el programa Win32DiskImager, se seleccionará el archivo descomprimido anterior y se instalará el sistema operativo, en el caso de este proyecto en la tarjeta SD que estará conectada al PC. Finalmente se hace clic sobre 'Write' y en unos minutos la instalación habrá acabado.

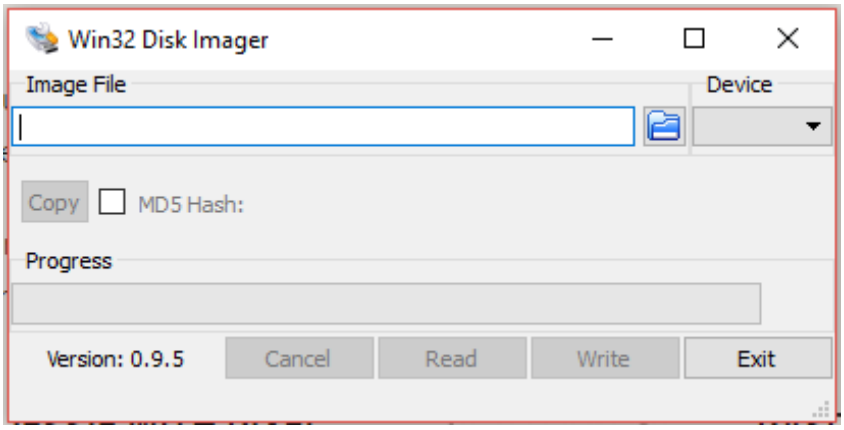


Fig. 4.12. Imagen programa Win32 Disk Manager. Fuente: Propia

Finalmente se debe introducir la tarjeta SD en la ranura de la Raspberry Pi destinada a ello.

4.4.2. Putty y VNC

Una vez instalado el sistema operativo, la Raspberry Pi 3 está lista para usarse. El acceso y control de la Raspberry Pi 3 se puede llevar a cabo de diversas formas.

El método más sencillo para conectarse a la Raspberry Pi 3 es conectando un teclado y un ratón a los puertos USB de ésta y una pantalla mediante HDMI.

También se puede acceder a la Raspberry Pi 3 mediante acceso remoto. De esta manera se puede dotar de autonomía al robot y así no tener que conectarse mediante cables.

Para poder conectarse remotamente a la Raspberry Pi 3 desde el PC se utilizarán dos programas: Putty [10] y VNC Vierwer [11].

Putty es un cliente SSH, Telnet, rlogin, y TCP raw con licencia libre desarrollado por un grupo de voluntarios. Es software beta, escrito y mantenido principalmente por Simon Tatham, de software libre y licenciado bajo la Licencia MIT.

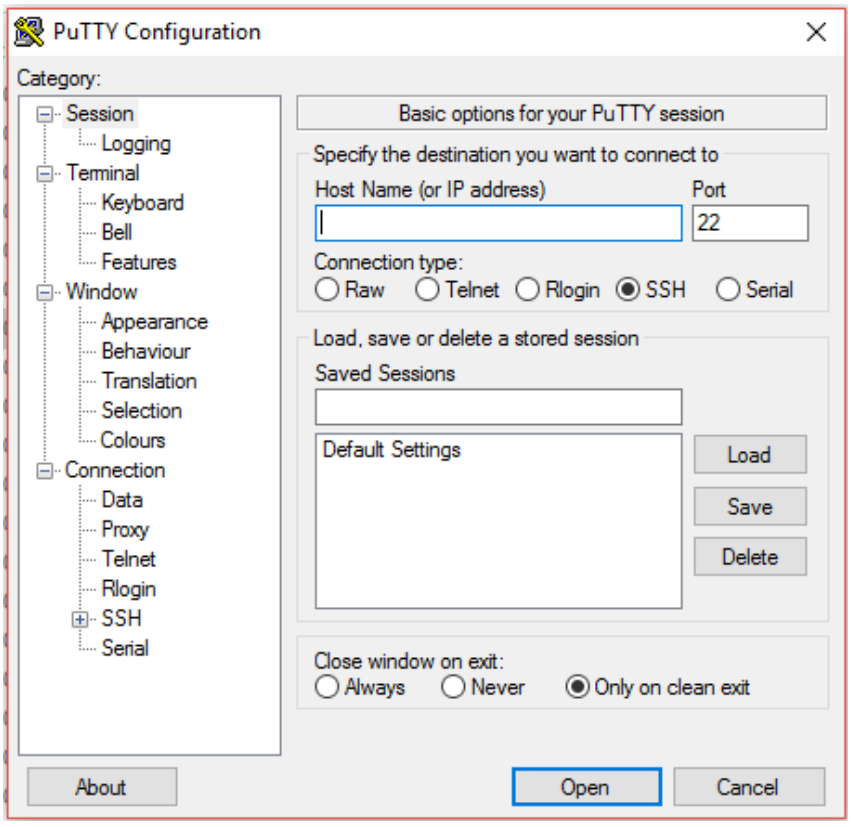


Fig. 4.13. Imagen Programa Putty. Fuente: Propia

En este proyecto se ha optado por el cliente SSH. SSH (Secure SHell, en español: intérprete de órdenes seguro) es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora mediante un intérprete de comandos, y también puede redirigir el tráfico de X (Sistema de Ventanas X) para poder ejecutar programas gráficos si tenemos ejecutando un Servidor X (en sistemas Unix y Windows)

A partir de este programa se puede acceder desde cualquier PC al intérprete de comandos de la Raspberry Pi. Para ello se debe ingresar la dirección IP de nuestra Raspberry Pi que deberá estar conectada a la misma red, ya sea mediante wifi o ethernet, que el PC desde la cual se controla.

Una forma fácil de trabajar, para no tener que buscar la dirección IP en cada conexión a la Raspberry Pi 3, es asignándole una dirección IP estática. Con ello se consigue que siempre sea la misma y por lo tanto la conexión sea mucho más rápida y sencilla.

El programa Putty permite acceder al intérprete de comandos de la Raspberry Pi 3, pero para poder trabajar más cómodamente e intuitivamente se necesita el acceso al escritorio y el entorno para poder visualizar, crear, eliminar o editar los archivos con un simple clic. Para ello se utilizará el programa VNC.

VNC es un programa de software libre basado en una estructura cliente-servidor el cual permite tomar el control del ordenador servidor remotamente a través de un ordenador cliente. También llamado software de escritorio remoto, VNC no impone restricciones en el sistema operativo del ordenador servidor con respecto al del cliente: es posible compartir la pantalla de una máquina con cualquier sistema operativo que soporte VNC conectándose desde otro ordenador o dispositivo que disponga de un cliente VNC portado.

Una vez se accede mediante Putty al intérprete de comandos, antes de abrir VNC por primera vez, es necesario escribir en el terminal:

```
sudo apt-get install tightvncserver
```

Con esta línea se instalará el servidor VNC en la Raspberry Pi 3. Por último, antes de abrir la aplicación VNC cada vez que se vaya a utilizar se deberá escribir en el terminal:

```
sudo vncserver :1 -geometry 1325x700 -depth 24
```

Con esta línea se inicializará el servidor VNC y ya se podrá utilizar el programa para acceder remotamente al escritorio de la Raspberry Pi 3.

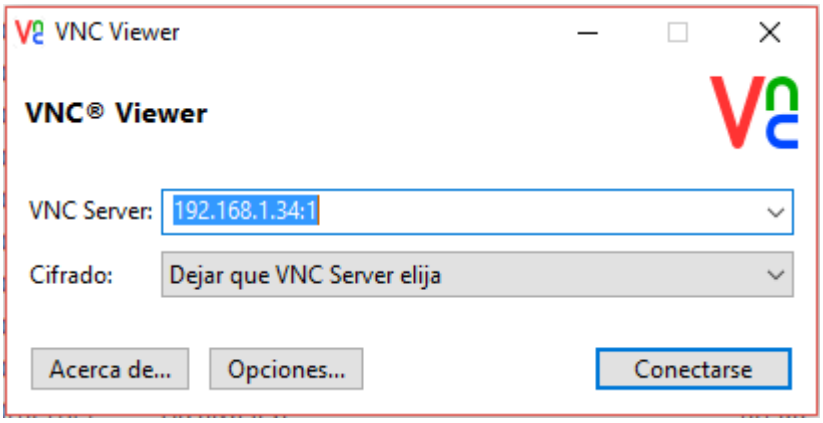


Fig. 4.14. Imagen programa VNC. Fuente: Propia

4.4.3. Blynk

Blynk es una plataforma que permite controlar placas como Arduino, Raspberry Pi, entre otras, a través de internet con una App de iOS o Android mediante un teléfono móvil inteligente.

Es una dashboard digital donde se puede construir una interfaz gráfica para un proyecto con botones simples de 'arrastrar y soltar', deslizadores y otros widgets y gráficos. También da la posibilidad de recopilar y visualizar datos de sensores presentes en el proyecto.

Blynk es una plataforma intuitiva y sencilla de utilizar, diseñada para el Internet de las Cosas.

Fig. 4.15. Imagen aplicación Blynk. Fuente: blynk.cc

La plataforma está formada por tres componentes:

- **App:** se debe descargar desde Play Store si se utiliza un móvil Android o desde App Store si se utiliza el sistema operativo iOS de Apple. En ambos casos se trata de una aplicación móvil donde se pueden crear diversos proyectos que incorporan diferentes widgets y gráficos a elección del usuario.

- **Servidor:** es el responsable de todas las comunicaciones entre el terminal móvil y el hardware del proyecto. Se puede utilizar un servidor público llamado Blynk Cloud o si se requiere de más privacidad uno privado localmente.

- **Librerías:** a disposición del usuario hay diversas librerías con las cuales se permite la comunicación entre dispositivos de una manera clara e intuitiva.

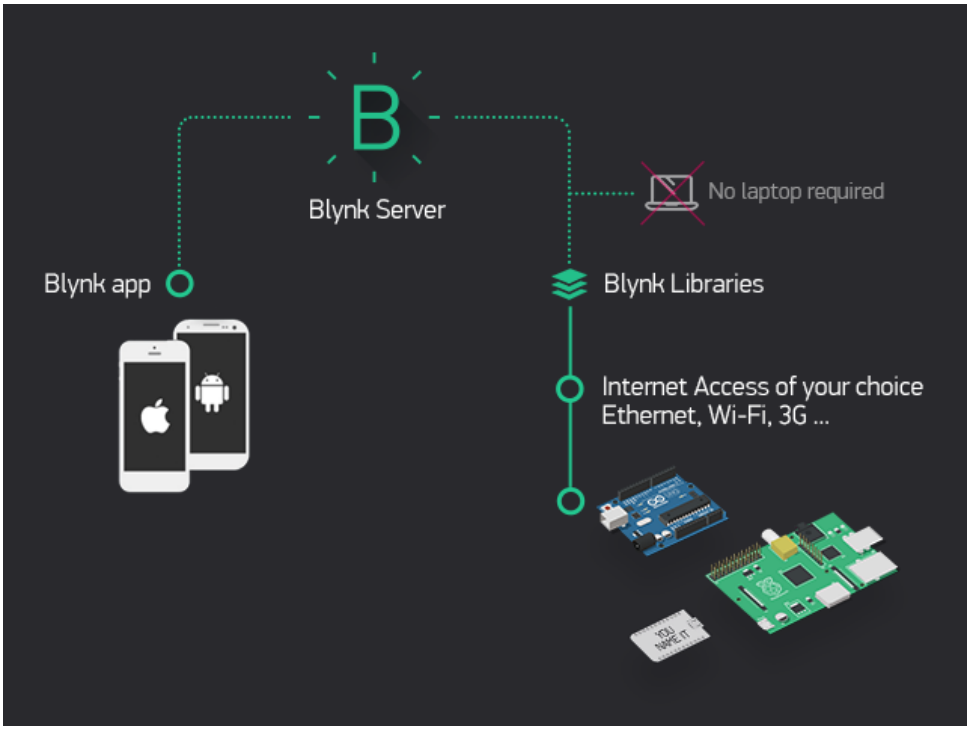


Fig. 4.16. Diagrama plataforma Blynk. Fuente: blynk.cc

A partir de este diagrama podemos observar el funcionamiento de la Plataforma Blynk. La información fluye en ambos sentidos, es decir, puede salir del terminal móvil y llegar al hardware del proyecto pasando por el servidor y viceversa. Uno de los puntos fuertes de la plataforma es que no es necesario un PC para este intercambio de información.

Para poder empezar a utilizar la aplicación Blynk hay que seguir un conjunto de pasos descritos en su página web [3]. Entre ellos figura como crear una cuenta para acceder al servidor y guardar proyectos, como crear un proyecto y como iniciarlo en el controlador utilizado.

En el caso concreto de este proyecto se han utilizado cinco widgets para controlar los servomotores y el electroimán. En la figura 4.17 se puede observar la interfaz gráfica de la aplicación Blynk con los diferentes widgets que se utilizarán en el control del brazo robótico.

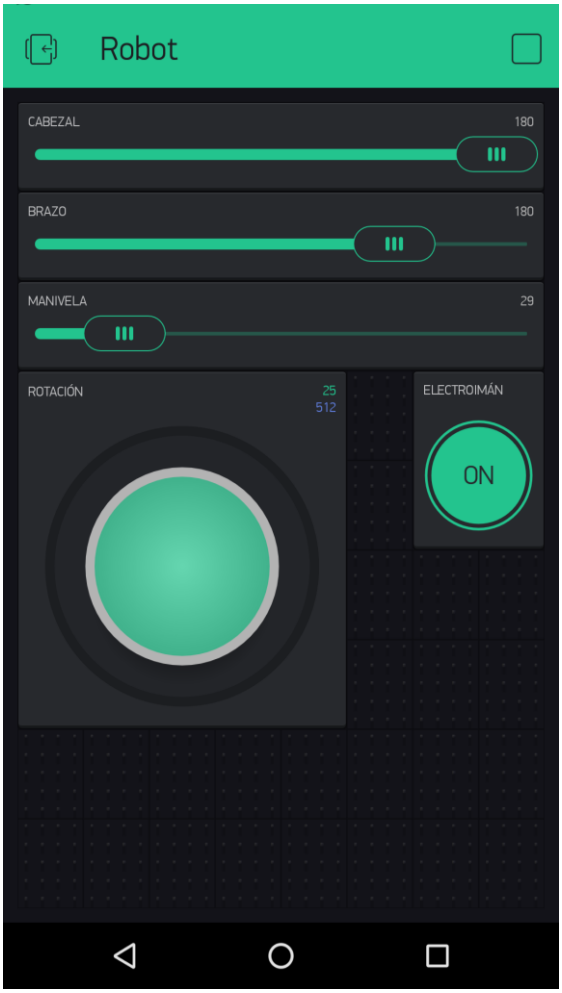


Fig. 4.17. Interfaz aplicación Blynk. Fuente: Propia

Tres de estos widgets son ‘deslizadores’, o también llamados sliders en inglés, y controlan los servos del cabezal, manivela y brazo. Como se puede observar en la figura 4.18 cada deslizador está conectado a un Pin Virtual (1, 2, 3) y posteriormente se explicará cómo funcionan.

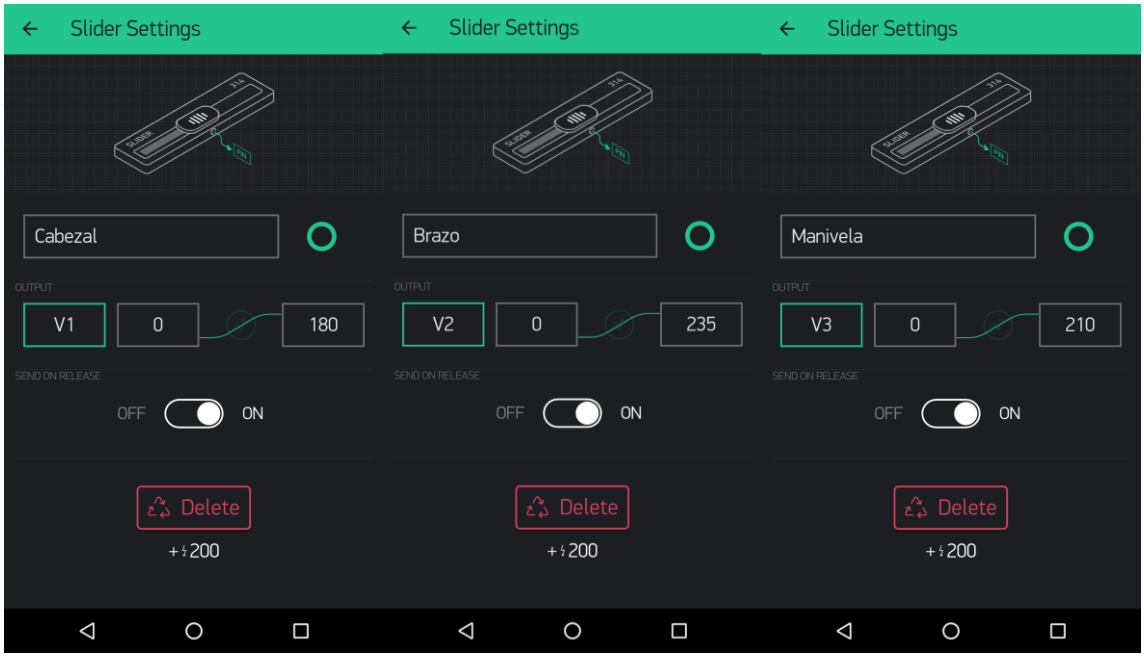


Fig. 4.18. Imagen aplicación Blynk (Pines virtuales sliders). Fuente: Propia

El servomotor de rotación continua se controla con una ‘palanca de mando’, o también llamado joystick en inglés, en el cual solo utilizamos uno de sus dos ejes: el horizontal, conectado al pin virtual 11.

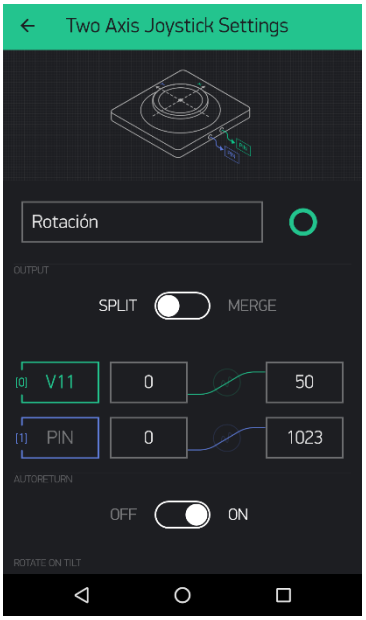


Fig. 4.19. Imagen aplicación Blynk (Rotación). Fuente: Propia

Se utiliza este tipo de widget dado que si no hay contacto con la pantalla la esfera virtual retorna al centro y el servomotor deja de girar. Si se mueve la esfera hacia la derecha el servomotor gira en el sentido de las agujas del reloj proporcionalmente a la distancia del centro. Sin embargo, si se mueve la esfera hacia la izquierda, el servomotor gira en el sentido contrario a las agujas del reloj.

Por último, para controlar el electroimán se utiliza un botón conectado al pin virtual 10. Pulsándolo se activará o se desactivará el electroimán según convenga.

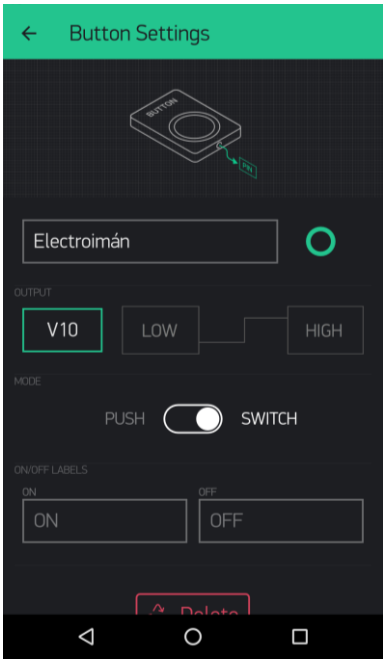


Fig. 4.20. Imagen aplicación Blynk (Electroimán). Fuente: Propia

Una vez creado el proyecto en el teléfono móvil hay que preparar la Raspberry Pi 3 para que se conecte con Blynk. Para ello hay que escribir las siguientes líneas en la consola de comandos de la Raspberry Pi:

```
sudo apt-get update && sudo apt-get upgrade

sudo apt-get install build-essential

sudo npm install -g npm

sudo npm install -g onoff

sudo npm install -g blynk-library
```

Con estas líneas se consigue actualizar la Raspberry Pi e instalar una serie de paquetes necesarios para el uso de la librería de Blynk. La última línea está destinada a instalar la librería mencionada.

4.4.4. Programa

En este apartado se presenta y se explica el software necesario para el correcto funcionamiento del robot.

A lo largo del grado en ingeniería en tecnologías industriales en la ETSEIB se han impartido varias asignaturas de informática y en todas ellas se ha utilizado el lenguaje Python. Es por ello que desde un principio la idea era utilizar este lenguaje pensando en la posibilidad de la creación de una asignatura similar a Proyecto I ó II, donde los alumnos podrían entender con mayor facilidad el programa. Sin embargo, la plataforma Blynk, actualmente, no ofrece la posibilidad de programar en Python y por ello se ha escogido un lenguaje alternativo.

Todo el software del proyecto se ha programado en el lenguaje JavaScript. Se ha escogido este lenguaje porque es compatible con la plataforma Blynk y además es muy similar al lenguaje Python, el cual es conocido por los estudiantes del grado en cuestión.

Librerías

Antes de comenzar a diseñar el código, que se utilizará para el control de los servos mediante la placa PCA9685 y el electroimán mediante la Raspberry Pi, hay que escoger e instalar las librerías de las placas controladoras que mejor se adecuen a las necesidades.

Además de la librería Blynk mencionada anteriormente se ha instalado la librería adafruit-pca9685 [12] y rpio [13].

Adafruit-pca9685 es una librería de Node.js para controlar la placa PCA9685 con la Raspberry Pi.

Su instalación es sencilla, ya que solo se debe abrir la consola de comandos de la Raspberry Pi y escribir el siguiente comando escrito a continuación:

```
npm install adafruit-pca9685
```

Para poder utilizar la librería en el programa se debe añadir esta línea de código:

```
var makePwm = require( "adafruit-pca9685" );
```

Mediante esta línea se hace un llamamiento a la librería y ahora ya se podría crear una instancia de PWM usando:

```
var pwm = makePwm();
```

Después del anterior comando se crea un objeto de PWM que tiene cuatro métodos:

```
pwm.setFrequency(Frecuencia, FactorCorrección);
```

```
pwm.setPwm(canal, on, off);
```

```
pwm.setPulse(canal, pulso);
```

```
pwm.stop();
```

El primer método tiene la función de indicar la frecuencia del PWM, pero en el caso concreto de este proyecto no se utilizará, ya que por defecto esta frecuencia es de 50Hz, que es óptima para los servomotores.

Los dos siguientes métodos tienen una función similar, la creación del PWM, pero en el primero se puede especificar en qué momento comienza y acaba el pulso, on y off, y en el segundo solo se informa de la anchura del pulso. En ambos casos el primer argumento es el canal, donde hay que escoger uno de los 16 existentes de la placa PCA9685.

El último método corta la señal PWM, ya que si no se informa de lo contrario la señal cuadrada sigue un bucle infinito mientras haya alimentación.

RPIO es una librería de alto rendimiento de Node.js que proporciona acceso a la interfaz GPIO Raspberry Pi, así como I²C, PWM, y SPI.

De la misma manera que las anteriores librerías su instalación se hace mediante el siguiente comando:

```
npm install rpio
```

Para poder utilizar la librería en el programa se debe añadir esta línea de código:

```
var rpio = require('rpio');
```

En esta librería existen un gran número de métodos, ya que además de crear PWM también se puede comunicar mediante I²C y SPI entre otras funciones. Es por ello que solo se describen los métodos que se han utilizado a lo largo del proyecto.

```
rpio.open( pin , rpio.INPUT);
```

Con este método se accede al pin seleccionado de la Raspberry Pi en modo lectura. Mediante el siguiente comando podremos leer los datos del pin seleccionado:

```
rpio.read( pin );
```

Para acceder a un pin en modo escritura se hará mediante la siguiente línea:

```
rpio.open( pin , rpio.OUTPUT, rpio.LOW ó rpio.HIGH);
```

Además de seleccionar el pin y el modo escritura en este método también hay que escoger si se quiere que por defecto el pin escogido tenga un valor lógico bajo (rpio.LOW) o alto (rpio.HIGH).

Una vez se haya accedido al pin, para cambiar de estado lógico, se hará mediante el siguiente comando:

```
rpio.write( pin, rpio.LOW ó rpio.HIGH);
```

Cuando se quieren hacer pausas entre comando y comando se utilizará la siguiente línea:

```
rpio.sleep( s ); / rpio.msleep( ms );
```

En el primer comando hay que indicar el tiempo de espera en segundos y en el segundo comando en milisegundos.

Una vez instaladas las librerías necesarias se procede a la programación del software para el control del brazo robótico.

Primeramente, hay que acceder a las librerías y crear las variables necesarias para poder conectar todos los componentes del sistema. A continuación se muestra la cabecera del programa diseñado donde se inician y se accede a los componentes.

```
var makePwm = require("adafruit-pca9685");  // acceso a la librería adafruit-pca9685
var pwm = makePwm();                        // creación objeto PWM
var B = require('blynk-library');           // acceso a la librería Blynk
var rpio = require('rpio');                 // acceso a la librería rpio
var cabezal;                                // creación variable cabezal
var rotacion;                               // creación variable rotacion
var brazo;                                  // creación variable brazo
var manivela;                               // creación variable manivela
var eleciman;                               // creación variable eleciman


var blynk = new B.Blynk('auth token');      // creación variable y asignación de token de autorización

function init() {                           // inicio función init

  cabezal = new blynk.VirtualPin(1);        // asignación de variable cabezal a pin virtual 1 Blynk
  rotacion = new blynk.VirtualPin(11);      // asignación de variable rotacion a pin virtual 11 Blynk
  brazo = new blynk.VirtualPin(2);          // asignación de variable brazo a pin virtual 2 Blynk
  manivela = new blynk.VirtualPin(3);       // asignación de variable manivela a pin virtual 3 Blynk
  eleciman= new blynk.VirtualPin(10);       // asignación de variable eleciman a pin virtual 10 Blynk
  rpio.open(12, rpio.OUTPUT, rpio.LOW);     // acceso a pin 12 Raspberry Pi en modo escritura
                                            // y nivel logico bajo
};

init();                                     // ejecución función init
```

Fig. 4.21. Código inicialización. Fuente: Propia

Una vez iniciado el sistema, el programa se divide en cuatro partes, que son el control de los servomotores, el control del electroimán, los límites dinámicos de movimiento y el control de velocidad de los servomotores brazo y manivela.

Electroimán

A continuación se muestra las líneas de código dedicadas al control del electroimán con sus respectivos comentarios.

```
eleciman.on('write', function(param) {      // inicio función eleciman en modo escritura
  var val = Number(param);                  // asignación del valor del pin virtual a variable val
  if (val == 1) {                           // condicional si valor es 1
    rpio.write(12, rpio.HIGH);              // asignación de valor lógico alto a pin 12
} else {                                    // caso contrario condicional
    rpio.write(12, rpio.LOW);               // asignación de valor lógico bajo a pin 12
}});
```

Fig. 4.22. Código electroimán. Fuente: Propia

Una vez iniciada la función eleciman se asigna el valor del pin virtual asignado al botón (10) en la aplicación Blynk. Este botón tiene dos estados apagado (0) y encendido (1). Cuando el estado sea 1, mediante la librería rpio, se activará el valor lógico alto del pin 12 de la Raspberry Pi y mediante el circuito diseñado y mencionado anteriormente el electroimán se activará.

Servomotor rotación continua

A continuación se muestran las líneas de código dedicadas al movimiento del servomotor de rotación continua con sus respectivas explicaciones.

```
41  rotacion.on('write', function(param) {      // inicio función rotacion en modo escritura
42    var value1 = parseInt(param);             // asignación del valor del pin virtual a variable value1
43    value1 = value1 + 304;                    // sumar 304 a value1
44    pwm.setPwm(1,0,value1);                   // crear pulso PWM en canal 1 de PCA9685 de 0 a value1
45  });
```

Fig. 4.23. Código servomotor rotación continua. Fuente: Propia

Una vez iniciada la función *rotacion* en modo escritura se asigna el valor del pin virtual 11 del joystick o palanca de mando de la aplicación Blynk. Mediante la ecuación 4.7 se obtiene el ancho de pulso que hay que enviar a la placa PCA9685. El joystick se ha diseñado para que tenga un rango de 50 y al soltar la esfera retorne al centro que es 25. Es por ello que se suma 304 al valor, ya que si se aplica un pulso de anchura 329 (unidades ya adaptadas a la placa controladora) el servomotor está en reposo. El centro del joystick se hace coincidir con este punto para que al soltar la esfera el servomotor tenga una velocidad angular nula. En caso de que se quiera aumentar la velocidad de rotación habría que aumentar el rango del joystick, pero siempre teniendo en cuenta que debe quedar centrado en 329.

Estos números se han obtenido aplicando el método heurístico de ensayo y error. Se han ido probando diferentes tipos de pulsos para saber cuál es el que convenía para la aplicación.

Servomotor cabezal

A continuación se muestran las líneas de código dedicadas al movimiento del servomotor del cabezal con sus respectivas explicaciones.

```
35  cabezal.on('write', function(param) {       // inicio función rotacion en modo escritura
36    var value = parseInt(param);              // asignación del valor del pin virtual a variable value
37    var x = (value + 50)*2.1;                 // operaciones con value para obtener x
38    pwm.setPwm(0,0,x);                        // crear pulso PWM en canal 0 de PCA9685 de 0 a x
39  });
```

Fig. 4.24. Código servomotor cabezal. Fuente: Propia

Una vez iniciada la función cabezal en modo escritura se asigna el valor del pin virtual 1 del slider o deslizador de la aplicación Blynk. Dicho slider se ha diseñado para que tenga un rango de 180º simulando así el rango de grados de rotación que tiene. Para adecuar el pulso a cada

uno de los grados simulados se hace la operación descrita en el código. Una vez obtenido el pulso deseado se envía al servomotor mediante el canal 0 de la placa PCA9685.

Servomotor brazo y servomotor manivela

A continuación se muestran las líneas de código dedicadas al movimiento del servomotor del brazo con sus respectivas explicaciones. El control del servomotor manivela es muy similar cambiando una serie de valores explicados posteriormente.

```
52   brazo.on('write', function(param) {          // inicio función rotacion en modo escritura
53     value2 = parseInt(param);                  // asignación del valor del pin virtual a variable value2
54     value2 = value2 + 175                      // operacion variable value2
170    var bra2 = value2;                         // asignacion variable value2 a variable auxiliar
171    if (bra1 < bra2) {                         // condicional control de velocidad
172      while (bra1 != bra2) {                   // bucle control de velocidad
173          bra1 = bra1 + 1;                     // aumento varibale en 1 unidad
174          pwm.setPwm(2,0,bra1);                // crear pulso PWM en canal 2 de PCA9685 de 0 a bra1
175          rpio.msleep(5);                      // esperar 5 ms para volver a comenzar bucle
176   }} else {                                   // caso contrario
177      while (bra1 != bra2) {                   // bucle control de velocidad
178          bra1 = bra1 - 1;                     // disminución varibale en 1 unidad
179          pwm.setPwm(2,0,bra1);                // crear pulso PWM en canal 2 de PCA9685 de 0 a bra1
180          rpio.msleep(5);                      // esperar 5 ms para volver a comenzar bucle
```

Fig. 4.25. Código servomotor brazo. Fuente: Propia

Una vez iniciada la función brazo en modo escritura se asigna el valor del pin virtual 2 del slider o deslizador de la aplicación Blynk. Dicho slider se ha diseñado para que tenga un rango de 235 (unidades ya adaptadas a la placa controladora según la ecuación 4.7) simulando así la anchura máximo del pulso al que puede llegar el brazo en cualquier configuración. Para adecuar el pulso al rango mínimo al que puede trabajar el servomotor del brazo se le suma 175. Por lo tanto, el rango de anchura de pulsos en el que trabaja este servomotor es de 175 a 410. Como se puede observar en la figura, de la línea 54 a la 170 hay un corte. Es aquí donde están programados los limites dinámicos del servomotor que se explicaran posteriormente.

De la línea 170 a la línea 178 está programado el control de velocidad. Esta programación es necesaria, ya que el servomotor tiene mucha potencia y había riesgo de vuelco. Este software funciona de la siguiente manera:

La variable '*bra1'* es la posición inicial o la posición del brazo antes de mover el slider para llegar a otra configuración. Al ejecutar el programa esta variable tiene un valor por defecto de 220.

La variable '*bra2*' es la posición final a la cual se quiere llegar. Si la posición final es mayor que la inicial la variable bra1 va aumentando una unidad y se va enviando este pulso al servomotor mediante un bucle hasta que ambas variables son iguales. En cambio, si la

posición final es menor que la inicial ésta va disminuyendo en una unidad hasta llegar a la final. Para que este proceso funcione a cada bucle descrito se le aplica un retraso de 5 milisegundos.

El servomotor manivela está programado de la misma manera, pero el ancho de los pulsos de este va de 130 hasta 340. A este servomotor también se le aplica el control de velocidad del servomotor brazo cambiando las variables '*bra1*' y '*bra2*' por '*man1*' y '*man2*'.

Límites dinámicos

La existencia de este concepto en el proyecto es debida a que hay que limitar los movimientos de los servomotores para que en ninguna configuración el cabezal impacte contra el suelo o evitar las configuraciones en las que los servomotores sufran o produzcan esfuerzos en la estructura del brazo robótico.

Estos límites dinámicos se han aplicado tanto en el servomotor manivela como en el servomotor brazo. Para ello se han dividido el rango de valores de anchuras de los pulsos que pueden adoptar cada uno de los dos y se ha dividido en pequeños rangos de 10 unidades. En cada uno de estos pequeños grupos, mediante el método de ensayo y error, se han estipulado el rango de valores óptimos del otro servomotor. Por lo tanto, si uno de los dos servomotores está en una configuración determinada, el otro tendrá unas ciertas restricciones. Si el segundo servomotor sobrepasará el valor límite, el software actúa y convierte este valor indeseado en el límite óptimo más cercano.

A continuación se muestra una pequeña parte del código dedicado a los límites dinámicos del servomotor del brazo según la configuración del servomotor de la manivela.

```
55      if (value3 < 135) {
56        if (value2 > 375) {
57            value2 = 375;
58    }} else if (value3 >= 145 && value3 < 155) {
59        if (value2 > 370) {
60            value2 = 370;
61    }} else if (value3 >= 155 && value3 < 165) {
62        if (value2 > 355) {
63            value2 = 355;
64    }} else if (value3 >= 165 && value3 < 175) {
65        if (value2 > 350) {
66            value2 = 350;
67    }} else if (value3 >= 175 && value3 < 185) {
68        if (value2 > 375) {
69            value2 = 375;
70    }} else if (value3 >= 185 && value3 < 195) {
71        if (value2 >375) {
72            value2 = 375;
73        } else if (value2 < 180) {
74            value2 = 180;
75    }} else if (value3 >= 195 && value3 < 205) {
76        if (value2 >375) {
77            value2 = 375;
78        } else if (value2 < 185) {
79            value2 = 185;
```

Fig. 4.26. Código límites dinámicos. Fuente: Propia

La variable '*value3*' es la posición del servomotor manivela y '*value2*' la posición del servomotor brazo. Mediante condicionales se clasifica la posición del servomotor manivela en uno de los grupos de 10 unidades. Una vez está asignada la posición del servomotor manivela en el grupo correspondiente, si la variable '*value2*' es mayor que el límite superior o es menor que el límite inferior, ésta se iguala a dicho límite.

Este mismo proceso se repite en el código dedicado al servomotor manivela cambiando la variable '*value3*' por '*value2*' y viceversa, estableciendo los límites correspondientes.

5. Presupuesto

A continuación se detallan los costes del proyecto. Se han dividido en costes derivados de los componentes eléctricos y electrónicos utilizados, costes de recursos humanos y costes de equipamiento usado a lo largo del proyecto con su debida amortización.

5.1. Hardware y Componentes

En la siguiente tabla se muestra todo el hardware y componentes que conforman el proyecto detallando el proveedor, cantidad, coste unitario y coste total.

Descripción	Proveedor	Cantidad	Coste Unitario (€)	Total (€)
Servomotor MG995 180º	Diotronic	2	14,52	29,04
Servomotor SM2309S 180º	Diotronic	1	7,08	7,08
Servomotor S4303R 360º	Diotronic	1	15,25	15,25
Electroimán C-6092	Diotronic	1	5,87	5,87
Convertidor TMR0512	Diotronic	1	11,97	11,97
Placa de topos	Diotronic	1	2,15	2,15
Diodo 1N4004	Diotronic	Pack 10	0,47	0,47
Batería Ideus 6600	Mediamarkt	1	24,99	24,99
Raspberry Pi 3	Diotronic	1	49,25	49,25
Mosfet IRL2505	Diotronic	1	4,18	4,18
Controlador PCA9685	Adafruit (Internet)	1	14,95	14,95
Resistencia 100Ω	Diotronic	Pack 5	0,6	0,6
Resistencia 100kΩ	Diotronic	Pack 5	0,6	0,6
Estaño	Diotronic	1	1,25	1,25
Cables hembra-hembra	Diotronic	Pack 10 x 2	1,14	2,28
Cables hembra-macho	Diotronic	Pack 10	1,69	1,69
Cables USB	Diotronic	2	2,85	5,7
Bridas	Diotronic	Pack 50	1,32	1,32
Tarjeta Micro SD 8GB Sony	Mediamarkt	1	7,73	7,73
Total				186,37

Tabla 5.1. Costes de Hardware y Componentes. Fuente: Propia

5.2. Recursos Humanos

En la siguiente tabla se puede observar el tiempo dedicado por un ingeniero al proyecto en las diferentes fases de este. También se detallan el coste de cada fase y el total.

Descripción	Tiempo (h)	Coste (€/h)	Total (€)
Estudio previo e investigación	80	20	1600
Diseño del software	175	40	7000
Pruebas	30	40	1200
Total	285		9800

Tabla 5.2. Costes de Recursos Humanos. Fuente: Propia

En la fase de estudio previo e investigación, además del tiempo invertido en encontrar una solución óptima al proyecto, se incluye también el tiempo destinado a la adquisición y pruebas con el hardware. También se incluye el estudio y comprensión de ambas placas controladoras y la interconexión entre todos los componentes.

En la fase de diseño del software, además de diseñar toda la programación de la Raspberry Pi, también se ha destinado tiempo a escoger y comprender las librerías utilizadas. Como se ha comentado con anterioridad en la ETSEIB el lenguaje de programación utilizado es Python. Es por ello que también se ha empleado tiempo en analizar y estudiar JavaScript, lenguaje utilizado en la programación del proyecto.

La fase de pruebas está muy ligada a la fase de diseño del software, ya que mediante el método de prueba y error se ha ido programando el software del brazo robótico.

5.3. Equipamiento

En este apartado se tiene en cuenta el coste de todo el equipamiento y herramientas utilizadas a lo largo del proyecto.

En la siguiente tabla podemos observar el equipamiento utilizado detallando su coste, su amortización y el coste que conlleva al proyecto.

Descripción	Coste (€)	Vida útil (años)	Amortización	Coste real (€)
Ordenador	600	5	0,1	60
Soldador	10	6	0,08	0,8
Pelacables	3,5	10	0,05	0,17
Osciloscopio	583	10	0,01	5,83
Multímetro	19,75	6	0,08	1,58
Total				68,38

Tabla 5.3. Costes de Equipamiento. Fuente: Propia

5.4. Presupuesto Final

En la siguiente tabla se muestra un resumen con todos los gastos y costes del proyecto cuya suma hace un total de 10.054,75 €.

Descripción	Coste (€)
Hardware y componentes	186,37
Recursos humanos	9800
Equipamiento	68,38
Total	10.054,75

Tabla 5.4. Coste Total. Fuente: Propia

Como se puede observar la mayor parte de los recursos irían destinados a la partida de recursos humanos, ya que son muchas horas de dedicación de un ingeniero.

Por otra parte, cabe comentar que los programas o software utilizado son de código libre, por lo tanto no suponen un sobrecoste al proyecto.

6. Impacto Ambiental

Los componentes eléctricos y electrónicos utilizados en este proyecto están sujetos a la normativa básica de restricción de sustancias peligrosas RoHS [14]. La directiva RoHS restringe el uso de seis materiales peligrosos en la fabricación de varios tipos de equipos eléctricos y electrónicos. Estas son las seis substancias químicas que prohíbe:

- Plomo
- Mercurio
- Cadmio
- Cromo VI (Cromo hexavalente)
- PBB
- PBDE

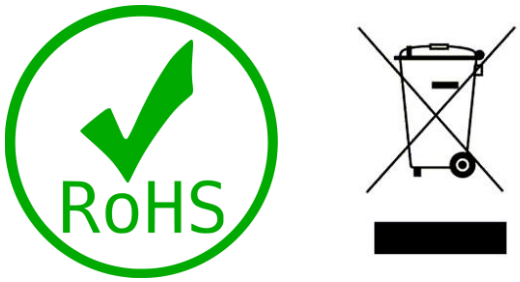


Fig. 6.1. Certificado RoHS y WEEE. Fuente: Google.es

Esta directiva está muy relacionada con la directiva de Residuos de Equipos Eléctricos y Electrónicos (WEEE por sus siglas en inglés). La directiva WEEE [15] es clave en la lucha por la conservación del medioambiente. Dada la masiva proliferación de componentes eléctricos y electrónicos con una vida útil muy corta debida a la continua innovación y creación de componentes nuevos se producen una inmensa cantidad de residuos cada año. La WEEE establece que los productores deben ser los responsables de asumir los costes de gestión de los residuos generados.

En el caso concreto de este proyecto los elementos más contaminantes son las placas controladoras y la batería. En el caso de las placas de circuito impreso su reciclaje consiste en la extracción de chips y la eliminación de metales mediante ácidos y calor. El reciclaje de las baterías de ion Litio, sin embargo, es más complejo, donde se extraen los metales existentes. El gran problema de estas baterías es que el Litio se desecha debido a su bajo precio 6€/kilo. Este metal es tóxico tanto para el medio ambiente como para el ser humano.

En conclusión, esto se traduce en emisiones a la atmósfera y vertido de metales a los suministros de aguas superficiales y subterráneas, que en altas concentraciones puede ser perjudicial para el medioambiente y el ser humano.

Conclusiones

Se han alcanzado los objetivos de este proyecto planteados en la introducción y especificaciones de este documento. Se ha conseguido diseñar y programar un software que controla los cuatro servomotores y el electroimán del brazo robótico, y se ha diseñado e implementado la electrónica necesaria para su correcto funcionamiento. Además, se ha documentado todo este proceso y se ha explicado paso a paso.

Se cumplen los objetivos secundarios definidos tales como, control del brazo robótico remoto con un teléfono inteligente mediante wifi, alimentación mediante una batería recargable y la adquisición de los componentes en comercios cercanos siempre intentando minimizar los costes.

Una vez terminado este proyecto, se plantean vías de mejora tanto en el software como en el hardware y mejoras en cuestiones de funcionalidad.

- Una opción de mejora sería la implementación de unos sensores o potenciómetros para saber en cualquier momento la posición y configuración en la cual se encuentra el brazo robótico.

- Para evitar el enredo de los cables de alimentación de las 2 placas controladoras, cuando gira el servomotor de rotación continua, se podría añadir una cadena portacables.

- En cuanto a software, se podría diseñar un código que indicando la posición en la cual se encuentra un objeto y a la cual se quiere llevar lo hiciera automáticamente, sin tener que manipular uno mismo los servomotores con la aplicación móvil.

- También se podrían mejorar los límites dinámicos del robot, implementado un algoritmo automático de dichos límites en función de la posición.

- En el cabezal se pueden hacer grandes mejoras, desde añadir una pinza con movimiento hasta incorporar una cámara digital para poder implementar aplicaciones de visión por computador.

Por último cabe comentar que esta memoria puede ser utilizada como una guía explicativa de la electrónica del brazo robótico en el caso de que finalmente éste se utilizara como material didáctico en alguna asignatura de la ETSEIB.

Agradecimientos

Para acabar me gustaría dedicar unas palabras a las personas que me han apoyado durante el proyecto y que han ayudado a tirar el proyecto hacia delante.

Primero me gustaría agradecer al tutor Manuel Moreno Eguílaz, por aportar su idea como TFG y por toda la ayuda prestada a lo largo de todo el proyecto. Sobretodo destacar su disponibilidad, implicación e interés mostrado para guiarme y corregirme siempre que lo he necesitado.

También me gustaría agradecer a mi compañero Guillem Ferré por las horas compartidas y dedicadas a que este proyecto pudiera cumplir con todos sus objetivos tanto mecánicos como electrónicos.

Para acabar me gustaría agradecer a mi familia y amigos por su interés en este proyecto.

Bibliografía

Referencias bibliográficas

[1] GUILLEM FERRÉ. *TFG: Construcció d'un micro-braç articulat: Part mecànica.* Barcelona: ETSEIB

[2] UFACTORY. *UArm, Milions of Possibilities,UFactory, 2015.* [https://ufactory.cc/en/, 20 de junio de 2016].

[3] BLYNK. *Plataforma para el control de microcontroladores.* [http://www.blynk.cc/, 20 de junio de 2016].

[4] ADAFRUIT. *Placa controladora de servomotores* [https://www.adafruit.com/product/815, 15 de mayo de 2016]

[5] FRITZING. *Open-source hardware initiative.* [http://fritzing.org/home/, 15 de setiembre de 2016]

[6] WIKIPEDIA. *Mosfet* [https://es.wikipedia.org/wiki/Transistor_de_efecto_de_campo_metal-%C3%B3xido-semiconductor, 10 de setiembre de 2016]

[7] DIOTRONIC. *Catálogo de diodos*. [http://www.diotronic.com/bolsa-10-diodos-smd-1a-400v_3491/, 15 de setiembre de 2016]

[8] RASPBERRY PI. *Sistema Operativo Raspbian.* [https://www.raspberrypi.org/downloads/raspbian/, 15 de mayo de 2016]

[9] PIDORA. *Sistema Operativo para Raspberry Pi.* [http://pidora.ca/, 15 de mayo de 2016]

[10] PUTTY. *Programa de control de remoto.* [http://www.putty.org/, 15 de mayo de 2016]

[11] REALVNC. *Programa de control remoto.* [https://www.realvnc.com/, 15 de mayo de 2016]

[12] GITHUB. *Librería control placa PCA9685.* [https://github.com/johntreacy/adafruit-pca9685, 1 de junio de 2016]

[13] NPM. *Librería de control PWM Raspberry Pi.* [https://www.npmjs.com/package/rpio, 1 de junio de 2016]

[14] COMISIÓN EUROPEA. *Normativa RoHS* [http://ec.europa.eu/environment/waste/rohs_eee/index_en.htm, 25 de setiembre de 2016]

[15] COMISIÓN EUROPEA. *Directiva WEEE.* [http://ec.europa.eu/environment/waste/weee/index_en.htm, 25 de setiembre de 2016]

Bibliografía complementaria

RASPBERRY PI 3 Datasheet. Raspberry Pi

PCA9685 Datasheet. Adafruit

IRL2505 Datasheet. International Rectifier

1N4004 Datasheet. Axial Lead Standard Recovery Rectifiers

TMR0512 Datasheet. Traco Power

Anexo

Programa

```
var makePwm = require("adafruit-pca9685");  // acceso a la librería adafruit-pca9685
var pwm = makePwm();                        // creación objeto PWM
var B = require('blynk-library');           // acceso a la librería Blynk
var rpio = require('rpio');                 // acceso a la librería rpio
var cabezal;                                // creación variable cabezal
var rotacion;                               // creación variable rotacion
var brazo;                                  // creación variable brazo
var manivela;                               // creación variable manivela
var eleciman;                               // creación variable eleciman


var blynk = new B.Blynk('auth token');      // creación variable y asignación de token de autorización

function init() {                           // inicio función init

  cabezal = new blynk.VirtualPin(1);        // asignación de variable cabezal a pin virtual 1 Blynk
  rotacion = new blynk.VirtualPin(11);      // asignación de variable rotacion a pin virtual 11 Blynk
  brazo = new blynk.VirtualPin(2);          // asignación de variable brazo a pin virtual 2 Blynk
  manivela = new blynk.VirtualPin(3);       // asignación de variable manivela a pin virtual 3 Blynk
  eleciman= new blynk.VirtualPin(10);       // asignación de variable eleciman a pin virtual 10 Blynk
  rpio.open(12, rpio.OUTPUT, rpio.LOW);     // acceso a pin 12 Raspberry Pi en modo escritura
                                            // y nivel logico bajo
};

init();                                     // ejecución función init

eleciman.on('write', function(param) {      // inicio función eleciman en modo escritura
  var val = Number(param);                  // asignación del valor del pin virtual a variable val
  if (val == 1) {                           // condicional si valor es 1
    rpio.write(12, rpio.HIGH);              // asignación de valor lógico alto a pin 12
} else {                                    // caso contrario condicional
    rpio.write(12, rpio.LOW);               // asignación de valor lógico bajo a pin 12
}});

cabezal.on('write', function(param) {       // inicio función rotacion en modo escritura
  var value = parseInt(param);              // asignación del valor del pin virtual a variable value
  var x = (value + 50)*2.1;                 // operaciones con value para obtener x
  pwm.setPwm(0,0,x);                        // crear pulso PWM en canal 0 de PCA9685 de 0 a x
});

rotacion.on('write', function(param) {      // inicio función rotacion en modo escritura
  var value1 = parseInt(param);             // asignación del valor del pin virtual a variable value1
  value1 = value1 + 304;                    // sumar 304 a value1
  pwm.setPwm(1,0,value1);                   // crear pulso PWM en canal 1 de PCA9685 de 0 a value1
});

var bra1 = 220                              // valor por defecto de variable
var man1 = 220                              // valor por defecto de variable
var value2 = 220                            // valor por defecto de variable
var value3 = 220                            // valor por defecto de variable

brazo.on('write', function(param) {         // inicio función brazo en modo escritura
  value2 = parseInt(param);                 // asignación del valor del pin virtual a variable value2
  value2 = value2 + 175                     // operacion variable value2
  if (value3 < 135) {                       // inicio limites dinámicos brazo
    if (value2 > 375) {
        value2 = 375;
}} else if (value3 >= 145 && value3 < 155) {
    if (value2 > 370) {
        value2 = 370;
}} else if (value3 >= 155 && value3 < 165) {
    if (value2 > 355) {
        value2 = 355;
}} else if (value3 >= 165 && value3 < 175) {
    if (value2 > 350) {
        value2 = 350;
}} else if (value3 >= 175 && value3 < 185) {
```

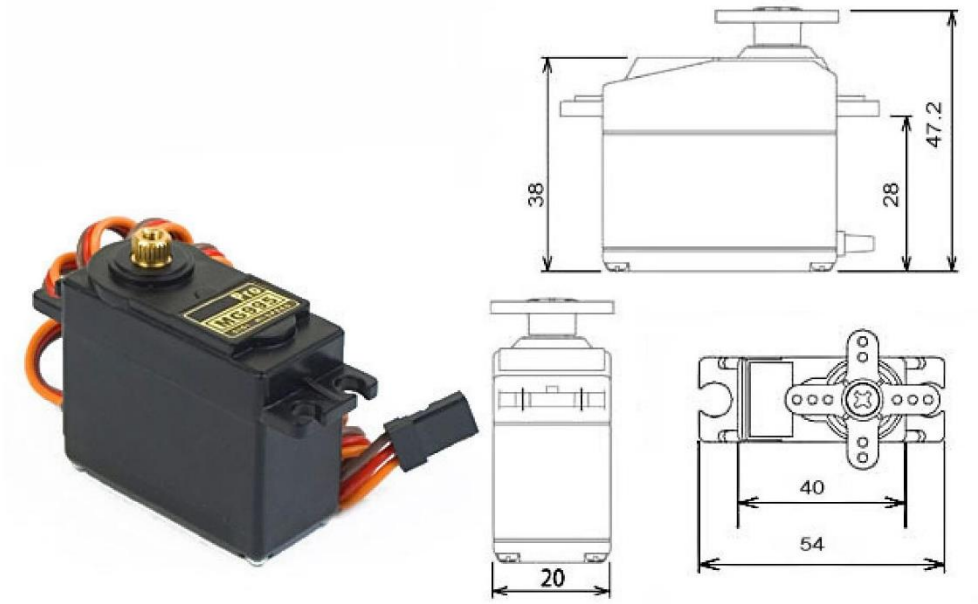
```
68      if (value2 > 375) {
69          value2 = 375;
70  }} else if (value3 >= 185 && value3 < 195) {
71      if (value2 >375) {
72          value2 = 375;
73      } else if (value2 < 180) {
74          value2 = 180;
75  }} else if (value3 >= 195 && value3 < 205) {
76      if (value2 >375) {
77          value2 = 375;
78      } else if (value2 < 185) {
79          value2 = 185;
80
81  }} else if (value3 >= 205 && value3 < 215) {
82      if (value2 >375) {
83          value2 = 375;
84      } else if (value2 < 190) {
85          value2 = 190;
86
87  }} else if (value3 >= 215 && value3 < 225) {
88      if (value2 >375) {
89          value2 = 375;
90      } else if (value2 < 195) {
91          value2 = 195;
92
93  }} else if (value3 >= 225 && value3 < 235) {
94      if (value2 >375) {
95          value2 = 375;
96      } else if (value2 < 200) {
97          value2 = 200;
98
99  }} else if (value3 >= 235 && value3 < 245) {
100     if (value2 >375) {
101         value2 = 375;
102     } else if (value2 < 205) {
103         value2 = 205;
104
105 }} else if (value3 >= 245 && value3 < 255) {
106     if (value2 >375) {
107         value2 = 375;
108     } else if (value2 < 210) {
109         value2 = 210;
110
111 }} else if (value3 >= 255 && value3 < 265) {
112     if (value2 >375) {
113         value2 = 375;
114     } else if (value2 < 215) {
115         value2 = 215;
116
117 }} else if (value3 >= 265 && value3 < 275) {
118     if (value2 >375) {
119         value2 = 375;
120     } else if (value2 < 220) {
121         value2 = 220;
122
123 }} else if (value3 >= 275 && value3 < 285) {
124     if (value2 >375) {
125         value2 = 375;
126     } else if (value2 < 220) {
127         value2 = 220;
128
129 }} else if (value3 >= 285 && value3 < 295) {
130     if (value2 >345) {
131         value2 = 345;
132     } else if (value2 < 1225) {
133         value2 = 1225;
134 }} else if (value3 >= 295 && value3 < 305) {
135     if (value2 >345) {
136         value2 = 345;
137     } else if (value2 < 230) {
138         value2 = 230;
139
140 }} else if (value3 >= 305 && value3 < 315) {
141     if (value2 >335) {
142         value2 = 335;
```

```
    } else if (value2 < 230) {
        value2 = 230;

}} else if (value3 >= 315 && value3 < 325) {
    if (value2 >325) {
        value2 = 325;
    } else if (value2 < 235) {
        value2 = 235;

}} else if (value3 >= 325 && value3 < 335) {
    if (value2 >295) {
        value2 = 295;
    } else if (value2 < 235) {
        value2 = 235;

}} else if (value3 >= 335 && value3 < 345) {
    if (value2 >295) {
        value2 = 295;
    } else if (value2 < 235) {
        value2 = 235;

}} else if (value3 >= 345 && value3 < 355) {
    if (value2 >295) {
        value2 = 295;
    } else if (value2 < 235) {
        value2 = 235;
}};                                  // final limites dinámicos brazo
  var bra2 = value2;                 // asignacion variable value2 a variable auxiliar
  if (bra1 < bra2) {                 // condicional control de velocidad
    while (bra1 != bra2) {           // bucle control de velocidad
        bra1 = bra1 + 1;             // aumento varibale en 1 unidad
        pwm.setPwm(2,0,bra1);        // crear pulso PWM en canal 2 de PCA9685 de 0 a bra1
        rpio.msleep(5);              // esperar 5 ms para volver a comenzar bucle
}} else {                            // caso contrario
    while (bra1 != bra2) {           // bucle control de velocidad
        bra1 = bra1 - 1;             // disminución varibale en 1 unidad
        pwm.setPwm(2,0,bra1);        // crear pulso PWM en canal 2 de PCA9685 de 0 a bra1
        rpio.msleep(5);              // esperar 5 ms para volver a comenzar bucle
}};
});
manivela.on('write', function(param) {      // inicio función brazo en modo escritura
  value3 = parseInt(param);                 // asignación del valor del pin virtual a variable value3
  value3 = value3 + 130;                    // operacion variable value3
  if (value2 < 180) {                       // inicio límites dinámicos manivela
    value3 = 130;
} else if (value2 < 190) {
    if (value3 >175) {
        value3 = 175;
}} else if (value2 < 200) {
    if (value3 >200) {
        value3 = 200;
}} else if (value2 < 210) {
    if (value3 >220) {
        value3 = 220;
}} else if (value2 < 220) {
    if (value3 >240) {
        value3 = 240;
}} else if (value2 < 230) {
    if (value3 >260) {
        value3 = 260;
}} else if (value2 < 240) {
    if (value3 >285) {
        value3 = 285;
}} else if (value2 < 250) {
    if (value3 >320) {
        value3 = 320;
}} else if (value2 >= 290 && value2 < 300) {
    if (value3 > 330) {
```

```
211             value3 = 330;
212     }} else if (value2 >= 300 && value2 < 310) {
213         if (value3 >320) {
214             value3 = 320;
215     }} else if (value2 >= 310 && value2 < 320) {
216         if (value3 >310) {
217             value3 = 310;
218     }} else if (value2 >= 320 && value2 < 330) {
219         if (value3 >300) {
220             value3 = 300;
221     }} else if (value2 >= 330 && value2 < 340) {
222         if (value3 >290) {
223             value3 = 290;
224     }} else if (value2 >= 340 && value2 < 350) {
225         if (value3 >280) {
226             value3 = 280;
227     }} else if (value2 >= 350 && value2 < 360) {
228         if (value3 > 240) {
229             value3 = 240;
230         } else if (value3 < 150) {
231             value3 = 150;
232     }} else if (value2 >= 360 && value2 < 370) {
233         if (value3 >230) {
234             value3 = 230;
235         } else if (value3 < 170) {
236             value3 = 170;
237     }} else if (value2 >= 370 && value2 < 380) {
238         if (value3 > 220) {
239             value3 = 220;
240         } else if (value3 < 190) {
241             value3 = 190;
242     }} else if (value2 >= 380 && value2 < 390) {
243         if (value3 >220) {
244             value3 = 220;
245         } else if (value3 < 190) {
246             value3 = 190;
247     }};                                     //final límites dinámicos manivela
248       var man2 = value3;                    // asignacion variable value3 a variable auxiliar
249       if (man1 < man2) {                    // condicional control de velocidad
250         while (man1 != man2) {              // bucle control de velocidad
251             man1 = man1 + 1;                // aumento varibale en 1 unidad
252             pwm.setPwm(3,0,man1);           // crear pulso PWM en canal 3 de PCA9685 de 0 a man1
253             rpio.msleep(5);                 // esperar 5 ms para volver a comenzar bucle
254     }} else {                               // caso contrario
255         while (man1 != man2) {              // bucle control de velocidad
256             man1 = man1 - 1;                // disminución varibale en 1 unidad
257             pwm.setPwm(3,0,man1);           // crear pulso PWM en canal 3 de PCA9685 de 0 a man1
258             rpio.msleep(5);                 // esperar 5 ms para volver a comenzar bucle
259     }}});
```

Especificaciones técnicas

MG995 High Speed
Metal Gear Dual Ball Bearing Servo

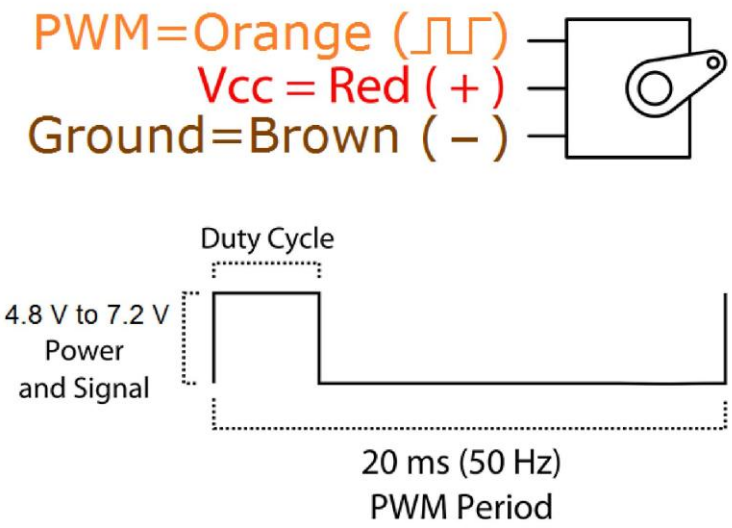


The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

This high-speed standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG995 Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 8.5 kgf·cm (4.8 V), 10 kgf·cm (6 V)
- Operating speed: 0.2 s/60º (4.8 V), 0.16 s/60º (6 V)
- Operating voltage: 4.8 V a 7.2 V
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC


PWM=Orange
Vcc = Red (+)
Ground=Brown (–)
Duty Cycle
4.8 V to 7.2 V
Power
and Signal
20 ms (50 Hz)
PWM Period

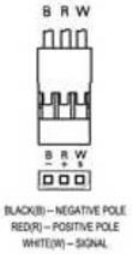
SM-S2309S MOTOR

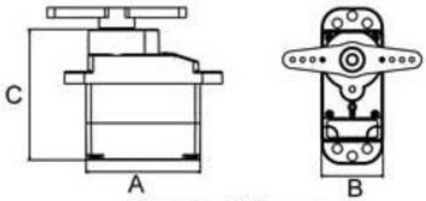
No. : SM-S2309S

Size: 22.9x12.3x22.2mm / 0.9x0.49x0.87in

Weight: 0.35oz

Specifications: Micro analog servo,4 plastic gears + 1 metal gear





Dimension（Unit：mm）

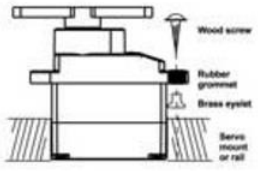


Diagram of installation

Products specification								Technical parameters						
Size(mm)					Weight		Wire	4.8V			6V			Rotation angle
								Speed	Torque		Speed	Torque		
A	B	C	D	E	g	oz	cm	sec/60°	kg•cm	oz•in	sec/60°	kg•cm	oz•in	
22.9	12.3	22.2	-	-	9.9	0.35	20.0	0.11	1.1	15.3	0.09	1.3	18.1	±60°

(Specifications are subjected to change without notice.)

Product brochure

Micro analog servo,4 plastic gear+ 1 metal gear.

Products packing

◦Packing with elevators (Elevators+PE bag)

Packaging content:Servo×1PCS、Servo arm×1bag、Manual×1PCS

Packaging specifications：Size-120×85mm、PE bag：120×85×0.07mm、Net weight-9.9g、Gross weight-11.5g

◦General packing (PE bag)

Packaging content：Packing with PE bag

Packaging specifications：PE bag size-90×85×0.07mm、Net weight-9.9g、Gross weight-11.5g

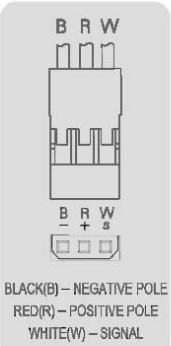
SPRINGRC®
www.springrc.com

43R Servo(360° Rotation) Specification

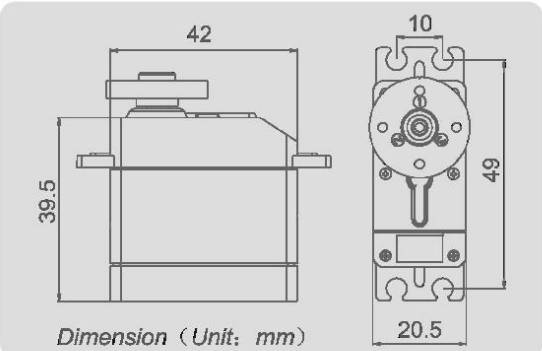
Thank you for choosing Spring Model's product

MODEL	TYPE	WEIGHT		4.8V			6V			DESCRIPTION	
		g	oz	SPEED	TORQUE		SPEED	TORQUE		GEAR	BEARING
				r/min	kg.cm	oz.in	r/min	kg.cm	oz.in		
SM-S4303R	Analog	44	1.55	60	3.3	45.8	70	4.8	66.7	1Metal Gear+ 4Plastic Gear	2
SM-S4306R		44	1.55	60	5.0	69.4	50	6.2	86.1	1Metal Gear+ 4Plastic Gear	2
SM-S4309R		60	2.12	58	7.9	109.7	49	8.7	120.8	Metal Gear	2
SM-S4315R		60	2.12	62	14.5	201.4	53	15.4	213.9	Metal Gear	2

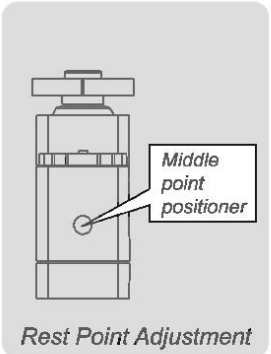
▲ 43R Robot series servo controled via analog signal(PWM),stopped via middle point positiner.
▲ Standard interface(like JR)with 30cm wire.
▲ Rotation and Rest Point Adjustment:when analog signal inputs,servo chooses orientation according to impulse width.when intermediatevalue of impluse width is above 1.5ms, servo is clockwise rotation,conversely,anticlockwise.Rest point need use slotted screwdriver to adjust the positioner carefully.Servo stopped rotation when the input signal is equivalent to impluse width.
▲ Please choose correct model for your application.
Caution: Torque over-loaded will damage the servo's mechanism.
▲ Keep the servo clean and away from dust, corrosive gas and humid air.
▲ Without further notification when some parameters slightly amend for improving quality.



BLACK(B) – NEGATIVE POLE
RED(R) – POSITIVE POLE
WHITE(W) – SIGNAL



Dimension（Unit：mm）



Rest Point Adjustment

SPRING MODEL ELECTRONICS Co.,LTD. MADE IN CHINA XBBS4306R119 V1.0

RoHS CE

ETSEIB

cebek®

SOLENOID / electromagnet
SOLENOIDE / ELECTROAIMANT
SOLENOIDE / ELECTROIMAN
C-6092

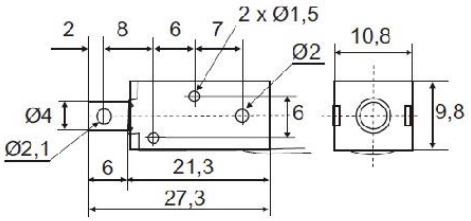
TECHNICAL CHARACTERISTICS

Power: 1.1 W
Voltage: 12V
Resistance: 131W
Current: 92mA
Time "on": indefinite
Amper x return: 198 (at 20 ° C)
Winding 2162 turns
Connection: 2 UL1571 AWG28 wires 100mm
Total weight: 10g
Piston Weight: 2g

Open weave electromagnet.
Thanks to its optimized design and use materials of high permeability and high-density windings, this magnet provides maximum strength with a very small size and weight. This model is suitable for professional and educational applications, such as: Locks, brakes, kicks, escapes, parts supply, positioning, robots, etc.. Operation: When you connect the electromagnet attracts the plunger inside.

Electroaimant avec couverture complète.
Grâce à sa conception optimisée et à ses matériaux de haute perméabilité ainsi que ses bobinages de haute densité, cet électroaimant offre la force maximale avec des dimensions et un poids très réduits.
Ce modèle est spécialement recommandé pour des applications professionnelles et éducatives, comme : serrures, freins, retours arrières, fournitures de pièces, positionnements, robots, etc... Fonctionnement: Lorsque vous connecter l'électroaimant, il attire le piston vers l'intérieur

Electroimán de armadura abierta.
Gracias a su optimizado diseño y a emplear materiales de alta permeabilidad y bobinados de alta densidad, este electroimán suministra la máxima fuerza con un tamaño y peso muy reducidos.
Este modelo es apto para aplicaciones profesionales y educativas, tales como:
Cerrojos, frenos, retrocesos, escapes, suministros de piezas, posicionados, robots, etc. Funcionamiento: al conectar el electroimán, atrae el émbolo hacia su interior.



Drawn in on position (plunger fully inserted)

Dimensions in mm

TECHNICAL SPECIFICATIONS

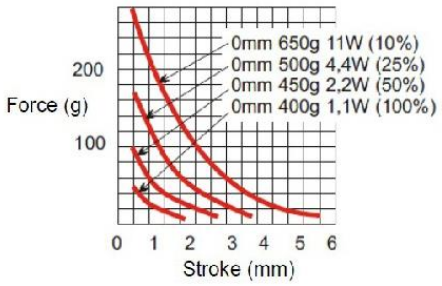
CYCLE(%): ON / (ON + OFF) x 100	100% continuous	50% (or less)	25% (or less)	10% (or less)
"ON" time maximum (seconds)	indefinite	50s	18s	2s
Power at 20 ° C	1,1W	2,2W	4,4W	11W
Amper x back to 20 ° C	198	281	396	627
voltage C.C	12V	17V	24V	38V

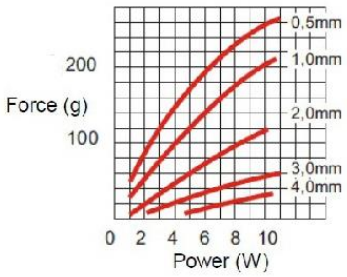
Race	Force
0 mm	400 g
0,5 mm	50 g
1 mm	22 g
2 mm	10 g

www.cebek.com - sat@cebek.com

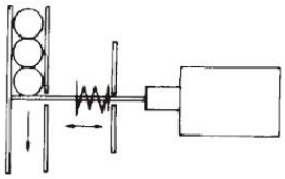
ETSEIB

C-6092

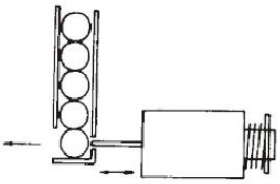




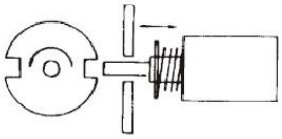
TYPICAL APPLICATIONS



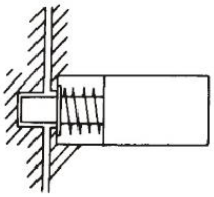
supply parts
Fourniture de pièces
Suministrar piezas



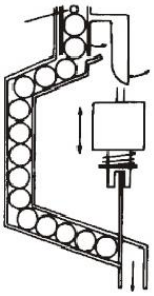
Provide "one on one"
Fourniture "une à une"
Suministrar " uno a uno"



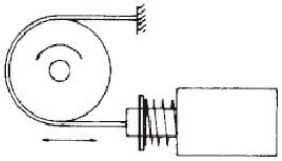
Position
Positionner
Posicionar



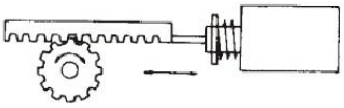
Locks / locks / latches
Serrures
Cerraduras / cerrojos / pestillos



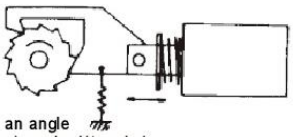
supply parts
Fourniture de pièces
Suministrar piezas



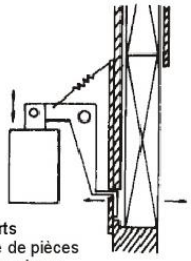
Brake
Freiner
Frenar



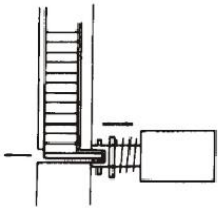
Back
Retour arrière
Retroceder



Rotate an angle
Tourner à angle déterminé
Girar un ángulo determinado



supply parts
Fourniture de pièces
Suministrar piezas



supply parts
Fourniture de pièces
Suministrar piezas